Content

INTRODUCTION **2**

OPEN SHOP SCHEDULING OF A LINUX CLUSTER USING MAUI/TORQUE **– PAPER BY MAARTEN DE RIDDER** **3**

PRIMARY RADAR PERFORMANCE ANALYSIS AND DATA COMPRESSION **- PAPER BY STIJN DELARBRE** **9**

MIGRATION OF A TIME-TRACKING SOFTWARE APPLICATION (ACTITIME) - **PAPER BY MAARTEN DEVOS** 14

WAN OPTIMIZATION CONTROLLERS RIVERBED TECHNOLOGY VS IPANEMA TECHNOLOGIES **- PAPER BY NICK GOYVAERTS** **19**

LINE-OF-SIGHT CALCULATION FOR PRIMITIVE POLYGON MESH VOLUMES USING RAY CASTING FOR RADIATION CALCULATION **– PAPER BY KAREL HENRARD** **24**

INTERFACING A SOLAR IRRADIATION SENSOR WITH ETHERNET BASED DATA LOGGER **- PAPER BY DAVID LOOIJMANS** **29**

CONSTRUCTION AND VALIDATION OF A SPEECH ACQUISITION AND SIGNAL CONDITIONING SYSTEM **- PAPER BY JAN MERTENS** **33**

POWER MANAGEMENT FOR ROUTER SIMULATION DEVICES **- PAPER BY JAN SMETS** **39**

ANALYZING AND IMPLEMENTATION OF MONITORING TOOLS (APRIL 2010) **- PAPER BY PHILIP VAN DEN EYNDE** **43**

THE IMPLEMENTATION OF WIRELESS VOICE THROUGH PICOCELLS OR WIRELESS ACCESS POINTS **– PAPER BY JO VAN LOOCK** **49**

USAGE SENSITIVITY OF THE SAAS-APPLICATION OF IOS INTERNATIONAL **– PAPER BY LUC VAN ROEY** **55**

FIXED-SIZE LEAST SQUARES SUPPORT VECTOR MACHINES STUDY AND VALIDATION OF A C++ IMPLEMENTATION **– PAPER BY STEFAN VANDEPUTTE** **60**

IMPROVING AUDIO QUALITY FOR HEARING AIDS **- PAPER BY PETER VERLINDEN** **66**

PERFORMANCE AND CAPACITY TESTING ON A WINDOWS SERVER 2003 TERMINAL SERVER **- PAPER BY ROBBY WIELOCKX** **72**

SILVERLIGHT 3.0 APPLICATION WITH A MODEL-VIEW-CONTROLLER DESIGNPATTERN AND MULTI-TOUCH CAPABILITIES **- PAPER BY GEERT WOUTERS** **78**

COMPARATIVE STUDY OF PROGRAMMING LANGUAGES AND COMMUNICATION METHODS FOR HARDWARE TESTING OF CISCO AND JUNIPER SWITCHES **– PAPER BY ROBIN WUYTS** **83**

Introduction

We are proud to present you this first edition 2009-10 of the Proceedings of M.Sc. thesis papers from our Master students in Engineering Technology: Electronics-ICT.

Sixteen students report here the results of their research. This research was done in companies, research institutions and our department itself. The results are presented as papers and collected in this text which aims to give the reader an idea about the quality of the student conducted research.

Both theoretical and application-oriented articles are included.

Our research areas are:

- Electronics
- ICT
- Biomedical technology

We hope that these papers will give the opportunity to discuss with us new ideas in current and future research and will result in new ways of collaboration.

The Electronics-ICT team

Patrick Colleman
Tom Croonenborghs
Joan Deboeck
Guy Geeraerts
Peter Karsmakers
Paul Leroux
Vic Van Roie
Bart Vanrumste
Staf Vermeulen

Open Shop Scheduling of a Linux Cluster using Maui/Torque

Maarten De Ridder[1], Gert Van Den Eynde[2], Radu Popescu[2], *Joan De boeck*[1]

[1] KHK Geel IIBT
Kleinhoefstraat 4
B-2440 Geel

[2] SCK•CEN Mol
Boerentang 200
B-2400 MOL

***Abstract*—Scheduling and information services are two main clusters, which play an important role in the overall performance of an application running on that cluster. When thinking of scheduling one might say that it is sufficient to make sure that the minimum requirements and QoS requirements are met for the execution of a job. Unfortunately, the whole process is much more complicated. First of all, a distinction will have to be made between serial jobs which are forced to run on only one processor on one hand, and parallel jobs which may use processors different nodes on the other hand. Secondly different company policies will have to be incorporated. Those company policies are not unambiguous but perhaps one of the most important elements of scheduling jobs that belong to different departments within a company. A third challenging problem encountered is the lack of uniformity. In order to develop a 'fair' system we need to acquire a legitimate information system to log the usage of each department, user etc., with the purpose to use that information for future scheduling decisions. Since the Ferm2 cluster, that will be using the scheduling system explained in this paper, has grown over many years and the fact that many different systems (i.e. nodes) are part of this cluster , it is intricate to compare node usage. In this paper we are going to discuss a manner of calculating a solution which approaches a optimal solution for a system. Furthermore we will explain how to overcome the lack of scheduler to distinct between different processors en use this distinction to correctly calculate de fairshare targets for different users of groups.**

I. Introduction

Parallel processing already existed in the minds of cpu-designers in 1940s and 1950s. Parallelism that would today be called horizontal microcode appeared in Turing's 1946 design of the Pilot ACE [1], and was carefully described by Wilkes [2]. Indeed, in 1952 Wilkes and Stringer wrote, "In some cases it may be possible for two or more micro operations to take place at the same time"[3].

nowadays parallel processing is being used to create more processing power. By using parallel processing one could highly increase the speed of execution. The idea behind this technology is the fact that a code and commands can be written in such a way that processes can be divided into smaller parts which then can be run simultaneously on different processors. It is as plain as day that as long as the process can be segmented into more and smaller pieces without creating too much overhead and interconnections, it will be better to use as many resources as possible. Because supercomputers were often quite expensive only large companies or institutes were able to acquire such processing power. The solution to this problem was clustering many low-end cpu's together. By doing so one could create a system with processing power that could be compared to supercomputers but the cost was considerable less.

The introduction of clusters within a workplace not only results in a gain of performance and memory but also in more complexity. Once multiple users started using the same resources, there had to be some sort of way of organizing the distribution of those shared resources by using algorithms. Those algorithms are comparable with the scheduling algorithms used in the kernel. The choice of a particular algorithm used by the kernel may favor one class of processes over another[4]. This is also the case when scheduling jobs in a cluster. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following:

- **CPU utilization**: to keep the cpu as busy as possible.
- **Throughput**: number of processes that complete per time unit.
- **Turnaround**: total time between submission of a process and its completion.
- **Waiting time**: amount of a process has been waiting in the ready queue.
- **Response time**: amount of time it takes from when a request has been submitted until the first response is produced.
- **Fairness**: Equal cpu time for each trhead.

Once we have found some parameters to assess different systems we must try to adjust the system to the real life situation within the company. In our case that situation is

very dynamic. When working on scientific research (i.e. nuclear energy and the effects on the environment), first there is a preparatory period where the user has to prepare his input file. During this time there is no need for cluster time. Then a model is constructed. This involves a lot of processor time. Afterwards they have to analyze the output data after the calculation. This means that the cluster usage of the users depend highly on the state of their project which means that the overall cluster usage is very dynamic.This makes it necessary to adjust the scheduling rules very. The solution to this problem should resemble a servo system where the input is continuously changing according to the output. The output in this case would be the data received from the scheduling software and resource manager.

We started our work by dividing the task into three subsystems. First there is the queuing system. All jobs will be put in a particular queue. By setting the parameters for each queue there can be made a first sifting. Then each of the queuing systems will be associated with a particular set of resources. The management of those sets of resources together with the decision making when it comes to which job has priority over the other job will be the second subsystem. The last subsystem will be responsible for adjusting the setup of the first two system so that the overall system will stay as close to the optimal setup as possible. This will be done by changing the targets set by the scheduler.

II. Preliminary study

A. *Fermi2 cluster*

First of all it's important to have some knowledge of the cluster and the system which is managing the cluster. The Fermi2 cluster has grown over the years by adding new nodes. This results in a cluster who consists momentarily out of three different types of nodes. First there were nodes with a Intel Pentium 4 3.0 gHz processor. These nodes were accompanied by nodes with a Intel Pentium 4 3.6gHz processor. The third kind of nodes are AMD Opteron 3.0 gHz nodes. In total this makes 48 nodes. Each node has 4 logical processors. In the case of the AMD Opteron, those 4 logical processors consist of 4 physical processors. The Intel Pentium 4 nodes only have 2 physical processors but use a Multi-threading technique.

B. *Scheduling software*

Scheduling refers to the way processes are assigned priorities in a priority queue. This assignment is carried out by software known as a scheduler. It consists of mainly, CPU,. Throughput, Turnaround, Waiting and Response[5]. The scheduling software that was chosen by SCK•CEN was Maui. Maui is a scheduler developed by the Maui supercomputing center as an alternative to the default Loadleveler scheduler in their IBM SP environment [6]. The Maui Scheduler is a policy engine which allows sites control over when, where, and how resources such as processors, memory, and disk are allocated to jobs. In addition to this control, it also provides mechanisms which help to intelligently optimize the use of these resources and generally manage the system [6]. The main advantage of using scheduling software is the easy of setting different settings by manipulating elementary objects. In Maui we can work with five different objects.

- Jobs: A job consists of one or more requirements each of which requests a number of resources of a given type.
- Nodes: A node is a collection of resources with a particular set of associated attributes.
- Policies: Policies are generally specified via a config file and serve to control how and when jobs start.
- Quality of Service: The term quality of service (QoS) refers to resource reservation control mechanisms. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance The Maui Scheduler allows administrators fine grain control over QoS levels on a per user, group and account basis.
- Reservations: A reservation can be made to ensure that a job with high priority will run as soon as possible.

The manner in which all these objects are managed can be described in several core algorithms [7]. First of all there is a priority algorithm which will calculate the overall priority of every job that has been submitted. The basis of the priority algorithm is a First In, First Out (FIFO) algorithm but this is enhanced through the combination of weighted factors.

Priority = serviceWeight * serviceFactor +
resourceWeight * resourceFactor +
fairshareWeight * fairshareFactor +
directspecWeight * directspecFactor +
targetWeight * targetFactor +
bypassWeight * bypassFactor.

Priority Component	Evaluation Metrics	Use
Service	Current queue time and expansion factor	Allows favoring jobs with lowest current scheduling performance (promotes balanced delivery of job queuetime and expansion factor service levels)
Requested Resources	Requested processors, memory, swap, local disk, nodes, and processor-equivalents	Allows favoring of jobs which meet various requested resource constraints (i.e., favoring large processor jobs counters backfills proclivity for smaller jobs and improves overall system utilization)
Fairshare	User, group, account, QoS, and Class fairshare utilization	Allows favoring jobs based on historical usage associated with their credentials
Direct Priority Specification	User, group, account, QoS, and Class administrator specified priorities	Allows political priorities to be assigned to various groups
Target	Current delta between measured and target queue time and expansion factor values	Allows ability to specify service targets and enable non-linear priority growth to enable a job to reach this service target
Bypass	Job bypass count	Allows favoring of jobs bypassed by backfill to prevent backfill based job starvation

Figure 1: different factors and their explanation

where each Weight value is a configurable parameter and each Factor is calculated from the subcomponents user, group, initial priority, QoS etc.

After Maui has prioritized the jobs in queue, it starts the jobs one by one stepping through the priority list until it reaches a job that cannot start. Because all jobs posses a start time and a wallclock limit, Maui can determine the finish time of the jobs in the queue. When it encounters a high priority job which cannot start, it can calculate the earliest time the necessary resources will be available to start that job. The Backfill algorithm makes use of this knowledge by checking whether there is a low priority job who can be scheduled without delaying the high priority job. On a typical system backfilling will increase system utilization by around 20% and improve average job turnaround time by even more [8].

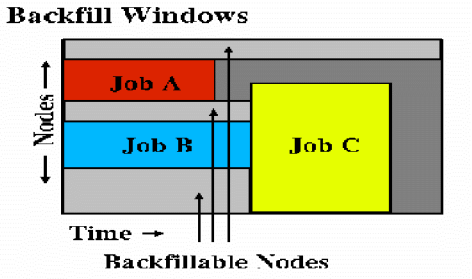


Figure 2: backfill example

In this case job A is a low priority job, job B and job C are high priority jobs. Even tough job C has a higher priority job A can start running because he does not interfere with job C starting time.

Another important feature is fairshare. There are a number of interpretations of the term fairshare as applied to batchsystems. In general, however, they each involve a mechanism which controls the distribution of delivered resources across various job attribute-based dimensions. They do this by tracking a utilization metric over time and using this historical data to adjust scheduling behavior so as to maintain resource usage within configured fairshare constraints [9]. Maui offers flexibility in configuring fairshare in areas including the tracked utilization metric, the utilization to job correlation attributes, the historical period, and the method of fairshare enforcement. Fairshare target usage can be specified on a per user, group, account, QOS, or class basis by way of a fairshare target. Each target is specified as a percentage value where each value is interpreted as a percent of delivered utilization. The use of delivered utilization as the target basis as opposed to using percent of configured or available resources allows the fairshare system to transparently take into account factors such scheduling inefficiencies, system maintenance, job failures, etc. The fairshare algorithm is composed of several parts. These parts handle tasks including the updating of historical fairshare usage information, managing fairshare windows, determining effective fairshare usage, and determining the impact of a job's various effective fairshare usage components. Unfortunately Maui does not have the ability to distinct between different cpu's. If we would relay completely on the fairshare capabilities of Maui then users who had used n Pentium 4 3.0 gHz processors would be penalized by the fairshare system as much as someone who had used n AMD Opteron processors. This would have a pernicious influence on the cluster usage because user would tend to wait for the faster nodes instead of running on slower nodes. To overcome this shortcoming we will be forced to change this externally.

C. *Resource manager*

Resources managers provide the low-level functionality to start, hold, cancel, and monitor jobs. Without these capabilities, a scheduler alone can not control jobs. In addition to Maui the Fermi2 cluster uses Torque as his scheduler. Torque is primarily used in batch systems. This means that once a job is submitted by a users Torque will handle everything from start to execution. Pooling resources in a batch system typically reduces technical administration of resources while offering a uniform view to users. Once configured properly, batch systems abstract away many of the details involved with running and managing jobs, allowing higher resource utilization. For example, users typically only need to specify the minimal constraints of a job and do not need to know the individual machine names of each host on which they are running. With this uniform abstracted view, batch systems can execute thousands and thousands of jobs simultaneously [10]. Torque uses four different components.

- Master Node - A batch system will have a master node where a job server runs. Depending on the needs of the systems, a master node may be dedicated to this task, or it may fulfill the roles of

other components as well.

- Submit/Interactive Nodes - Submit or interactive nodes provide an entry point to the system for users to manage their workload. For these nodes, users are able to submit and track their jobs. Additionally, some sites have one or more nodes reserved for interactive use, such as testing and troubleshooting environment problems. These nodes have client commands (such as qsub and qhold).
- Compute Nodes - Compute nodes are the workhorses of the system. Their role is to execute submitted jobs. On each compute node, pbs_mom runs to start, kill, and manage submitted jobs. It communicates with pbs_server on the master node. Depending on the needs of the systems, a compute node may double as the master node (or more).
- Resources - Some systems are organized for the express purpose of managing a collection of resources beyond compute nodes. Resources can include high-speed networks, storage systems, license managers, and so forth. Availability of these resources is limited and needs to be managed intelligently to promote fairness and increased utilization.

III. Queuing system

.

A. Introduction

In queuing theory the main objective is to manage the system is such a way that both the unit that has to wait and the one that serves get the most benefit [11]. The unit *demanding* service is identified as the customer. The unit providing service is known as the server. This terminology is used in a generic sense regardless of the nature of the physical context. In most cases the arrivals are a product of external factors. Therefore, the best one can do is to describe the input process in terms of random variables that can represent either the number arriving during a time interval or the time interval between successive arrivals. The uncertainties involved in the service mechanism are the number of servers, the number of customers being served at any time, and the duration and mode of service. Networks of queues consist of more than one server arranged in series and/or parallel. Random variables are used to represent service times, and the number of servers, when appropriate. If service is provided for customers in groups, their size can also be a random variable. The number of customers that can wait at a time in a queuing system is also a significant factor for consideration. If the waiting room is large, one can assume that, for all practical purposes, it is infinite. All other factors regarding the rules of conduct of the queue can be pooled under this heading. One of these is the rule followed by the server in accepting customers for service. In this context, rules such as "first come, first served'' (FCFS), "last come, first served'' (LCFS), and "random selection for service'' (RS) are self-explanatory. Others such as "round robin (RR)'' and "shortest processing time'' may need some elaboration. In many situations customers in some classes have priority in service over others.

B. Kendall Classification of Queuing Systems

This identification of these elements provides a taxonomy for symbolically representing with a variety of system elements [12]. The Kendall classification of queuing systems (1953) exists in several modifications. The most comprehensive classification uses 6 symbols.

A/B/s/q/c/p

the first characteristic A specifies the nature of the arrival process. Usually we assume that the interarrival times are independent and have a common distribution. The second characteristic B specifies the nature of the service time. They are also assumed to be independent and identically distributed and are independent of the interarrival times. The third characteristic s is the number of parallel servers. The fourth characteristic q describes the queue discipline. Depending on whether the customers are being treated FCFS, LCFS, etc. The fifth characteristic specifies the maximum allowable number of customers in the system, including customers who are waiting and customers who are in service (i.e. system capacity). The last characteristic gives the size of the population from which customers are drawn. Unless the number of potential customers is of the same order of magnitude as the number of servers, the population size is considered to be infinite. After analyzing the jobs who where submitted by the users the past few months, we could conclude that interarrival times and services times were independent, identically distributed random variables having an exponential distribution. Also the queue discipline used by SCK•CEN is non-preemptive Prioritizing thus the system we are working with can be described as M/M/c/Pnp/∞/∞. Before the upgrade from the cluster c would be 192 (48 nodes X 4 logical processors). Systems like these are called Markovian systems which stems from the fact that the exponential distribution is the only continuous distribution with the markov property. The Markov property states that the probability distribution for the system at the next step (and in fact at all future steps) only depends on the current state of the system, and not additionally on the state of the system at previous steps. i.e. it is memoryless [14].

C. Performance measures

Once the classification of our waiting queue is determined we further described the properties of the system. The common characteristic of all Markovian systems is that all interesting distributions, namely of the interarrival times and the distribution of the service times are exponential distributions and thus exhibit the markov property. From this property we find following important conclusions.

- The state of the system can be summarized in a single variable, namely the number of customers in the system.
- Markovian systems can be directly mapped to a continuous time markov chain (CTMC) which can then be solved.
- For Markovian systems the n-step transition probability $P_{ij}(n)$, which is the probability that after n transitions, a Markov chain will be in state j, given that the chain began in state I, will approach a limit π_j for large t which is independent of the initial state i. This limit is called the steady state or equilibrium probability of state j.

In our case, steady state means the probability that at an instant in the future, j jobs will be present in the system We can derive the steady state probability by using the flow balance equations or conservation equations for our system. Once the steady state is found, other important performance measures can be derived. One important factor is the utilization. CPU utilization is one of the factors used to make an assessment of the quality of one system against an other. An other parameter who is calculated is the average number of customers present in the queuing system L.

$$L = \sum_{j=0}^{j=\infty} j\pi_j$$

In this case we are interested in the expected number of jobs waiting in line (in queue) L_q. We can find L_q by deriving it from the steady state.

$$L_q = \sum_{j=0}^{j=\infty} (j-1)\pi_j = L - (1-\pi_0)$$

From the data extracted from the output files from the lasts months, it was possible to find the arrival rate λ and the service rate μ from the system. According to Little's law [15] we can say that the average time a job spends in the system W with

$$W = \frac{L}{\lambda}$$

Furthermore we can calculate the average time a job spends in line W_q by applying Little's law to L_q in stead of L.

D. Implementation

When it comes to maximizing the CPU utilization, this can best be done by setting the correct parameters for the scheduling software, in this case Maui. The main objective for the queuing system is to reduce the waiting time of the jobs. Therefore W_q will be used. We first divide the population of jobs into two subpopulations. The first subpopulations contains the serial jobs, the second subpopulations contains the parallel jobs. For each subpopulation we calculate W_q for different amounts of nodes available for that population. We now add W_{qsn} and $W_{qp(48-n)}$ where s and p indicate the type of the job and n equals the number of nodes used. The optimal solution can be found by finding n where the sum of W_{qsn} and $W_{qp(48-n)}$ is minimal. The reason why n was chosen to represent a node and not a processor is because the queues will be static and therefore one node cannot be part of two different queues. This same calculations has been done for subpopulations created by dividing the main population into 3 groups containing jobs from one department.

After discussing these calculations with the three departments and taking into account the company policy about the cluster usage for the different departments, we decided in favor of one high performance queue and using the rest of the nodes will be placed in dedicated queues. A high performance queue has the benefit of having few slots (approximately 4) but with each slot many nodes. This results in higher throughput. Even long lasting jobs can be handled within days or weeks. The disadvantage of a high performance queue is the lack of interactivity. In reality it may occur that a job contains a fault in the input file. In case of a fault the job will loose the reserved slot and will have to wait back in the queue. To prevent users from having to use a regular queue for testing purposes we will accommodate the queuing system with one small queue (i.e. few nodes per slot).

One consideration was made about the usage of serial jobs on the new nodes. Because the AMD Opteron nodes haven't got hyper-threading enabled, they are more suited for running serial jobs. This results from the fact that internode communication is much faster than intranode communication. Thus when mapping the queue to the nodes it seems best to reserve the AMD Opteron nodes for serial processing. This means that those departments that often submit serial jobs will get more AMD Opteron nodes and thus less Intel Core I7 nodes. To remain fair to all departments we ran a benchmark using a MCMP program, which will be used by most users to create their model. After finding the difference in speed between the AMD Opteron processor and a Intel Core i7 node we rearranged the amount of nodes each queue receives thus making sure every institute receives processing power according to their financial input.

IV. Scheduling algorithm

For scheduling purposes Maui is being used. At the moment the scheduling decisions are made by taking into account 3 parameters.

- Group configuration: by enabling the group priority the system can implement the company policy. Resources should be allocated according to the share in the investment.
- User configuration: Although this parameter is set, every user within one department has the same target and therefore everyone has equal priority. When a user is doing their internship at SCK•CEN this could be used to lower his priority.
- Fairshare configuration: the window of the fairshare is set to 15 which means that the last 15 days will be taken into account. Also the decay was set to 0.95 insuring that the recent past is more important but not fully decisive. The calculation for the fairshare target is depicted in figure 3.

Factor	Formula
Service	$\text{QueueTimeWeight} * \min(\text{QueueTimeCap}, \text{QueueTime}_J) + \text{XFactorWeight} * \min(\text{XFCap}, \text{XFactor}_J)$
Resource	$\text{MIN}(\text{RESOURCECAP}, \text{NODEWEIGHT} * \text{Nodes}_J + \text{PROCWEIGHT} * \text{Processors}_J + \text{MEMWEIGHT} * \text{Memory}_J + \text{SWAPWEIGHT} * \text{Swap}_J + \text{DISKWEIGHT} * \text{Disk}_J + \text{PEWEIGHT} * \text{PE}_J)$
Fairshare	$\text{MIN}(\text{FSCAP}, \text{FSUSERWEIGHT} * \text{FSDeltaUserUsage}[\text{User}_J] + \text{FSGROUPWEIGHT} * \text{FSDeltaGroupUsage}[\text{Group}_J] + \text{FSACCOUNTWEIGHT} * \text{FSDeltaAccountUsage}[\text{Account}_J] + \text{FSQOSWEIGHT} * \text{FSDeltaQOSUsage}[\text{QOS}_J] + \text{FSCLASSWEIGHT} * \text{FSDeltaClassUsage}[\text{Class}_J])$
Directspec	$\text{USERWEIGHT} * \text{Priority}[\text{User}_J] + \text{GROUPWEIGHT} * \text{Priority}[\text{Group}_J] + \text{ACCOUNTWEIGHT} * \text{Priority}[\text{Account}_J] + \text{QOSWEIGHT} * \text{Priority}[\text{QOS}_J] + \text{CLASSWEIGHT} * \text{Priority}[\text{Class}_J]$
Target	$(\text{MAX}(.0001, \text{XFTarget} - \text{XFCurrent}_J)^{-2} + (\text{MAX}(.0001, \text{QTTarget} - \text{QTCurrent}_J)^{-2}$ NOTE: XF is a unitless ratio while QT is reported in minutes.
Bypass	BypassCount_J

Figure 3: formula's for calculating different parameters

To make a prediction to which setup would be most optimal, we modeled the system using a simplex algorithm. Not only does this make an assumption about how to change the parameters, also a sensitivity analysis is possible to see which parameters can be altered an by how much.

Once the correct setup is found, most of the work is done by Maui and/of Torque. The only problem that remained was the lack of Maui to work with a non-uniform cluster. This could be resolved by making a script which will be run every day. This would mean that Maui had to be restarted every day to let the change in the setup file go through.

V. Conclusion

For most purposes the combination of using Maui and Torque to manage a linux cluster works well. The main issue is setting the different parameters in such a way that it resembles most the current real state of the cluster. By using queuing theory to become the steady state parameters of a system we can predict more or less how a system will behave over time and adjust the queuing system so to minimize the waiting time. After setting up the Maui setup file the system works adequate. Nonetheless will the usage of some clusters change dramatically over time. By adjusting the weight and factor of the different setup parameters the system can have a servo effect. In order to overcome the problem of a non-uniform cluster, we had to use a extern script to supplement

Maui so that the fairshare algorithm could cope with such a system. To do this Maui has to be restarted every time an

update of the fairshare target is being made. This is of coarse not preferable but nonetheless needful. Otherwise users of the cluster will try to overcome this themselves which is pernicious for the utilization of the cluster. Furthermore the system can accommodate continuously to changing usage of the cluster.

Acknowledgment

. I would like to thank my professor ing. Joan De Boeck of assisting me with this thesis. Also specials thanks go to Prof. dr. F.C.R. Spieksma; Faculty of Business and Economics KU Leuven and Prof.dr. Roel Leus, Faculty of Business and Economics KULeuven for pointing me in the right direction.

References

[1] B. E. Carpenter and R. W. Doran (Editor). A, M. Turin~'s ACE Report of 1946 and Other Papers. (MIT Press, Cambridge, Massachussetts, 1986).

[2] M. V. Wilkers, "The best way of designing a automatic calculating machine", Proc Manchester University Computer Conference 1951, pp. 16-18

[3] J.B. Stringer, "Microprogramming and the designs of the control circuits in electronic and digital computing", The Cambridge philosophical society, Part 2, 1953, pp. 230-238

[4] Avi Silberschatz , Peter Baer Galvin, Greg Gagne , "Operating System Concepts" John Wiley & Sons inc. 2004, isbn: 0-471-69466-5

[5] Prashant Shenoy, "Lectures on Operating Systems", university of Massachusetts, 2008, Lecture 7

[6] Maui Scheduler administration's guide ClusterResources chapter 1

[7] David Jackson, Quinn Snell, Mark Clement, "Core Algorithms of the Maui Scheduler", Brigham Young University lecture notes in computer science, Volume 2221/2001 pp 87-102

[8] Srividya Srinivasan, Rajkumar Kettimutu, Vijay Subrarnani, "Characterization of Backfilling. Strategies for Parallel Jobs", 2002 International Conference on Parallel Processing Workshops (ICPPW'02), 2002, pp. 514

[9] J. Kay, P. Lauder, "A fair scheduler", "Communications of the ACM" ACM New York, Volume 31, 1988, pp. 44-45

[10] Torque administrator's guide version 2.3, ClusterResources inc. http://www.clusterresources.com/

[11] U. Narayan Bhat, "An introduction to queuing theory", Birkhäuser Boston 2008, pp. 14-18, isbn: 978-0-8176-4724-7

[12] Shaler Stidham Jr., "Optimal Design of Queueing Systems", University of North Carolina, Chapel Hill North Carolina, Chapman & Hall/CRC Press 2009, ch. 1, isbn: 978-1-58488-076-9

[13] Wayne L. Winston, "Operations Research: Applications and Algorithms Fourth edition", Thomson Learning inc. - Brooks/Cole 2004 pp. 1061, isbn: 0-534-42362-0 .

[14] Andreas Willig, "A short Introduction to Queueing Theory", Technical University Berlin, Telecommunication Networks Group Sekr. FT 5-2, einsteinufer 25 Berlin, July 1999.

[15] Little, J. D. C. "A Proof of the Queueing Formula $L = \lambda W$" Operations Research, 9, 383-387 (1961).

Primary Radar Performance Analysis and Data Compression

S. Delarbre[1], N. Van Hoef[2], G. Geeraerts[1]

[1]IBW, K.H. Kempen (Associatie KULeuven), Kleinhoefstraat 4, B-2440 Geel, Belgium
[2]Intersoft Electronics nv, Lammerdries-Oost 27, B-2250 Olen, Belgium

stijn_delarbre@hotmail.com
niels.vanhoef@intersoft-electronics.com
guy.geeraerts@khk.be

***Abstract*—Research on radar performance is becoming more and more essential. It is important to assess radar performance based on calculated parameters and use these parameters to optimize or improve radar performance in certain situations. We will discuss real-time and offline radar parameter calculations in LabVIEW7 for future performance analysis based on primary radar raw video and secondary radar digital data. Secondly real-time compression of raw video data coming from primary radar using digital data from secondary radar in C++ and LabVIEW7 will be discussed. Raw video data compression may have its benefits. The smaller the data, the longer the recordings can be to fit on the same disk. It will turn out that data compression will speed up offline analysis and that disks will be used less intensively and less memory is needed. It will also become clear how certain parameters are implemented that lie on the basis of future performance analysis.**

I. Introduction

At present, radar systems are meant to run 24/7 and faults aren't always (immediately) detected. Most radar systems undergo maintenance on a monthly or tri-monthly basis and have to function at a reasonable performance all the time. Therefore, it is important to calculate radar parameters to assess radar performance. These parameters include:

- **Radar Cross Section (RCS):** RCS is used to assess radar sensitivity.
- **Signal-to-Noise ratio (SNR):** The higher the SNR the better a target can be recognized.
- **Pulse Compression:** Pulse compression processing gain will enhance detection and needs to be verified.
- **Parabolic Fit Error:** We try to fit a parabola in the slow time video return of a target. The difference between the slow time video return and the used parabola gives us an error number. Note that we use a parabola because a radar beam can be approximated by a parabola.
- …

These parameters can then be used to optimize or improve radar systems' performance. These calculated performance parameters could later on also be used to predict the performance of another (equivalent) radar system.

Offline radar system performance analysis was the first step taken to calculate the needed radar parameters. This way it was easy to check if the written algorithms work correctly and if they could be used in a real-time system. These algorithms could then be integrated in a real-time system together with a primary radar raw video data filter to filter useful data and analyze this data at the same time.

Real-time primary radar raw video data compression is, as mentioned above, another step taken. Data compression is important in the way of disk and memory usage. If we only write data to disk that is important for future analysis, there will be less memory taken and disks will be used less intensively. Of course it is also possible to analyze this data immediately after it is filtered. This way writing data to disk and analyzing data can be done at the same time. Another advantage of data compression is the reduction of read times afterwards which speeds up offline analysis, simply because there is less data to read. [1]

II. Data Representation

Before we can move on to calculation of radar parameters or data compression, it is important to take a look at how data is represented. We will therefore take a look at the representation of the two used data formats: primary radar raw video and secondary radar digital data.

A. Primary Radar Raw Video

Primary radar raw video consists of a byte stream where each two bytes (16 bits) represent one sample. The used data format is represented in Table 1.

TABLE 1
Primary Radar Raw Video Data Format

Sample 1	*Sample 2*	*Sample 3*	*Sample 4*
Analog 1 (12)	Analog 2 (12)	Analog 1 (12)	Analog 2 (12)
ARP (1)	1 (1)	0 (1)	1 (1)
ACP (1)	0 (1)	1 (1)	1 (1)
PPS (1)	Mode S (1)	0 (1)	Mode S (1)
Trigger (1)	Trigger (1)	Trigger (1)	Trigger (1)

This 16bit data is sampled at 16MHz using a RIM device (Radar Interface Module). Since I/Q data is interleaved, this is 8MSPS. [2] Analog 1 and Analog 2 represent 12bit I/Q data. The other 4 bits are digital bits where trigger, ACP and ARP (together with I/Q data) are important. The trigger bit is set when a new interrogation has started (when a new pulse is transmitted). The ACP (Azimuth Change Pulse) bit is set when the radar has rotated a given angle. Every time the ACP bit has been set, the ACP counter is incremented. The value of this counter is used to check where the radar is pointing at. The number of ACP pulses per rotation determines radar precision. A common value is 4096 which gives a radar precision of 0.087° per impulse. The ARP (Azimuth Reference Pulse) bit is set when the radar has reached a reference point (e.g. North). This pulse resets the ACP counter. [3]

We can use this byte stream to display the raw video in an intensity graph (Fig. 1), where the intensity represents target, clutter or noise power.

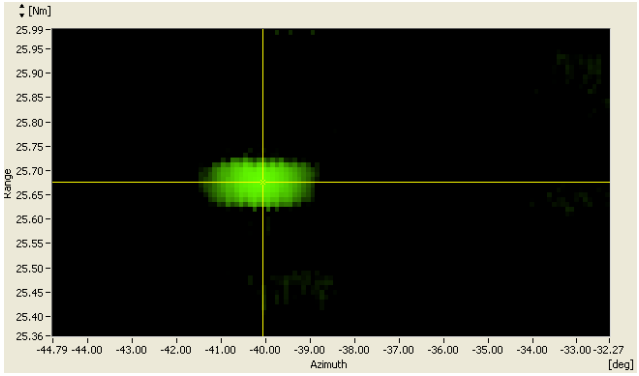


Fig. 1. Intensity graph of PSR Raw Video (single target)

B. *Secondary Radar Digital Data*

Digital data is stored in proprietary RASS-S6 data fields consisting of 128 bytes where each byte or set of bytes represents a property of the target. An example of a RASS-S6 data field is given in Figure 2.

3335345269,7422	Timestamp	6,231371	X (Nm)	0	FL. STAT
14	Target ID	-39,343330	Y (Nm)	0,000000	Æ Elev.
14	Track Nr	-300,000000	Velocity	0	Res 2
1	Scan Nr	0,000000	Heading	0	Res 3
◦0	1 Code	0	# PSR Refl.	0	Res 4
◦0	2 Code	0	# SSR Refl.	0	Res 5
◦0	A Code	×0	S-Address	0	Res 6
0	Altitude (Ft.)	0	MB Byte 1	256	Res 7
0	3D Heigth	0	MB Byte 2	0	Res 8
39,833752	Range	0	MB Byte 3	0	Res 9
0,000000	Æ Range	0	MB Byte 4	0	Res 10
171,000000	Azimuth	0	MB Byte 5		
0,000000	Æ Azimuth	0	MB Byte 6		
0	Plot Quality	0	MB Byte 7		
-62,65	Power	0	Com. Cap.		
×60400	Status	0	ACAS		

Fig. 2. RASS-S6 data field

The most important target properties in a RASS-S6 data field for us are:

- Scan Number
- Range
- Altitude (Ft.)
- Azimuth
- X (Nm)
- Y (Nm)

These properties are important because they allow us to track a target in the primary radar raw video. This makes it easy to calculate target/radar parameters which can then be used to analyze radar performance.

We can display each target (represented by a RASS-S6 data field) in an XY graph, where each plot represents one target return during one antenna revolution. An example of such an XY graph is shown in Figure 3.

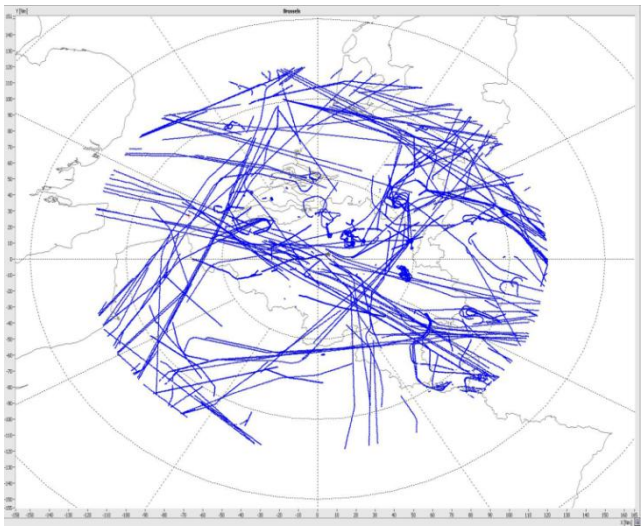


Fig. 3. XY graph of secondary radar digital data in LabVIEW7

III. Parameter Calculations

Now that we have understanding of the data representation, we can move on to radar parameter calculations. We will discuss RCS, parabolic fit error number and SNR calculation. All of these parameters are calculated using LabVIEW7. We will give an overview of what these parameters are, why they are important and how they are calculated. Note that when testing a radar system, we generate (perfect) targets with a RTG (Radar Target Generator) and inject these into the radar system so that radar performance only depends on the radar system itself. [4]

A. *Parabolic Fit*

Since a target's echo takes the form of a parabola in slow time, we can use parabolic fitting to calculate an error number that resembles the difference between the slow time video and a parabola (Fig. 4). This error number can then be used to assess radar performance.

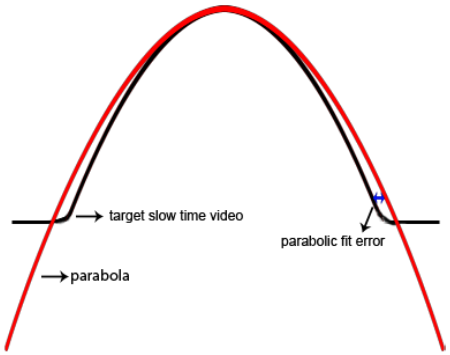


Fig. 4. Parabolic fit error of a target's slow time video return

Another use of parabolic fitting is locating a target. Since a target's slow time video return has a parabolic form it is easy to locate a target surrounded by noise using a parabolic fit (Fig. 5).

Calculation

Using the range and azimuth (or X and Y) from secondary radar data we are able to locate the target in raw video (Fig. 1). We will filter this target out of the raw video using a window. Next, we will take a look at each range in slow time as is shown in Figure 5.

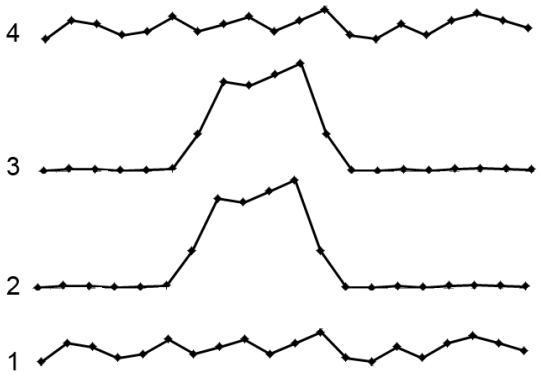


Fig. 5. Slow time raw video of a target

Each line in this figure represents slow time video at a certain range; the higher the number, the higher the range. If we now cross correlate each of these lines with a given parabola and calculate the maximum correlation for each line, we will have the best fit for the parabola with each of these given lines. Of course, it is easy to understand that lines 2 and 3 will have a better fit than lines 1 and 4. If we then compare these calculated maxima, either line 2 or 3 will have the maximum fit (suppose line 3). Next, we will calculate the line for which the maximum correlation is above half the correlation of line 3. This is done bottom up. The first line that meets this condition will be line 2. The range that corresponds to line 2 will be taken as the starting range of the target.

After calculating the starting range of the target, we will use a polynomial fit to calculate the target's azimuth location, power and an error number (mean squared error) between the target's slow time echo and the parabola that fits best as is shown in Figure 4.

The x-value and y-value of the calculated best fitting parabola's vertex will be used to represent respectively, the target's azimuth location and the amplitude (in Volts) of the target's reflected signal that is received by the antenna.

B. RCS

SKOLNIK [5] provides the following definition: *"The radar cross section of a target is the (fictional) area intercepting that amount of power which, when scattered equally in all directions, produces an echo at the radar equal to that from the target."*

RCS is used to assess radar sensitivity. It is used to measure the ability to detect a target at any given range. Targets with a low RCS like a Cessna might not be spotted at long range, while the new A380 which has a higher RCS will still be spotted. Of course at very long ranges, none of both planes will be spotted. RCS is a function of target range and received power at the antenna. [6][7]

Note that clutter plays a role in RCS calculations. Clutter is a term used for buildings, trees, surfaces, … that give unwanted echoes. When a target with a high RCS is in a low clutter area the target will be easily spotted. When the same target is located in an area where there is a lot of clutter, and the reflected power received back at the antenna coming from the clutter is equivalent to the power coming from the target, the target will be hard to spot or won't be spotted at all. [8][9]

We will therefore use secondary radar digital data to locate targets in raw video so that no targets will be lost due to clutter or no clutter will be seen as targets.

Calculation

We will first use the previously described parabolic fitting techniques to locate the target in the raw video and to calculate the amplitude of the target's reflected signal. We will then convert this voltage to decibels. This will give us the received power (P), in decibels, by the antenna coming from the target. Next, we are able to calculate the RCS of the target.

Calculating the RCS of a target consists of the following steps in our implementation (note that all parameters are represented in decibels):

1. First, the transmitted power is added to the antenna gain during transmission. This value is then subtracted from the target power P.
2. Next, path loss and extra influences, lens-effect and atmospheric attenuation, will be taken into account. These influences are calculated based on the elevation angle and range of the target. These influences will be added to the value obtained in step 1.
3. Third, the antenna gain during reception will be calculated and subtracted from the value calculated in step 2.
4. Finally, possible range, frequency and swerling influences are calculated and subtracted from the value calculated in step 3.

This will return a value in dBm^2 which is the RCS of the target. We can then use this value to predict at which locations the target will not be visible for the given radar system.

C. SNR

SNR or Signal-to-Noise Ratio is defined as the ratio of signal power to noise power. [10] SNR depends on target power, clutter and of course generated noise inside the radar system. We can use SNR to predict in which areas it will be hard to locate a target or to assess radar performance.

Calculation

As with RCS calculation, we will first use the previously described parabolic fitting techniques to locate the target.

Afterwards we will use fast time video (power-range) on the target's azimuth location to calculate the SNR as is shown in Figure 6.

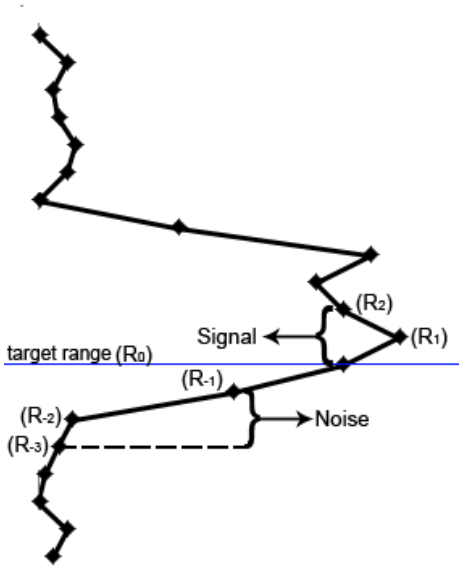


Fig. 6. Target fast time video

SNR is calculated using

$$SNR = \frac{\sum_{i=0}^{2} P_{R_i}}{\sum_{i=0}^{2} P_{R_{i-3}}} \quad (1)$$

where R represents range (Fig. 6) and P represents power (dB) at a certain range. The calculated SNR can then be used to predict target visibility at a certain range or in a cluttered area and to assess radar performance.

IV. Data Compression

Data compression is important in the way of disk and memory usage. If we only write necessary data to disk, data will take up less memory and disks will be used less intensively.

The speed of continuous writing is calculated using

$$R = f_s * N \quad (2)$$

where R represents write speed in MB/s, f_s represents sampling frequency in MHz and N represents the number of bytes per sample. Using a sampling frequency of 16MHz and having 2 bytes/sample, this gives us 32MB/s.

As shown, the write speed used for data writing without filtering is 32MB/s, which means that a 1TB disk will be full after recording about 9hours. If we exaggerate and state that there is only 1 target in unfiltered data on a 1TB disk, we have wasted about 99% of disk space, which is of course unwanted. If we then want to analyze the radar system, we will have to read all data and check all data for targets to analyze, which will both take up too much time. It could be, depending on the number of targets, that we are able to use a 1TB disk for a recording of 2 or more days, which is a big improvement. Therefore, filtering targets before writing raw video data to disk is a big step forward. We will do this by filtering a window out of primary radar raw video based on target information (range-azimuth) coming from secondary radar. This will not only improve disk usage, but it also speeds up the offline analyzing process.

Having shown the importance of data compression, we will give an overview of certain decisions taken during the process of writing the filtering program. These decisions have an influence on program complexity, disk/memory usage and determine the complexity of programs to read data afterwards.

Buffering

Buffering is the first important decision. Since secondary radar target information will not (always) reach the computer system at the same time the primary radar raw video of the same target does, it is important to buffer raw video for a certain time. Note that both primary and secondary radar are connected to the same PC/laptop.

The used buffer has to be large enough so that no data will be lost, but the buffer has to be small enough so that not all physical memory will be used for buffering. We have chosen the size of the buffer to fit 1 full scan of 360°. We have chosen this size because it is easy to work with and because simulations have shown that we won't lose any important data. The used buffer uses the FIFO algorithm. This means that the oldest data will be removed first, if necessary, when new data enters the buffer.

Threading

We had to take a decision concerning threading. If we would work with a single thread, we would have to check if there is a secondary radar target waiting to be filtered every time we run through the raw video coming from the primary radar. When using 2 threads, reading raw video will become independent from processing targets, thus execution becomes asynchronous. Therefore, when the execution of one of both threads lags, the other thread will keep executing in the correct way. For this reason, we have chosen to use 2 threads. Note that using 2 threads also makes our program easier to debug during implementation and easier to understand.

One thread maintains the buffer that contains the primary radar raw video and creates a list of what is inside the buffer. A second thread checks if there are targets waiting for filtering, and if there is a waiting target, it filters this target out of the buffered raw video. Of course both threads require some kind of synchronization so that no faulty data is filtered. [11] In other words, the second thread has to run fast enough so that no data is lost or wrong data is filtered. Simulations have confirmed that without any synchronization mechanism data is filtered in the right way.

Writing targets to disk

How we are going to write a target to disk is the last very important decision. It determines the complexity of the program, it has an influence on memory usage and it determines how we are going to read data afterwards.

We could create one index file in which every target's header is located and one data file or we could create a header for each target and attach the target's data to his header so we only have one file. We have chosen for the second option, because it is easier to program and it is easier to read data afterwards. When a target is filtered, its header is created and his raw video data is added. We then place this data (incl. header) into a second buffer which hands this data over to a second program that writes this data to disk.

V. Real-time Simulation/Experiment

Since we didn't have the possibility to test the real-time program at a radar station, we have written a program in LabVIEW7 that simulates a real-time system for 1 full scan. We use generated primary radar data and matching secondary radar data. Since synchronizing and simulating data streams in LabVIEW7 isn't an easy thing to do, we had to add some code in the real-time C++ program for testing purposes only.

Simulations have confirmed the working of the real-time filter and parallel calculation of the parabolic fit error number and SNR as described previously.

VI. Acknowledgements

We would like to express our gratitude to Peter Lievens for his technical support concerning Labview7. We would also like to express our gratitude to Erik Moons and Johan Vansant for their technical support concerning C++.

VII. Conclusions and Future Work

In this paper we have discussed radar parameter calculations which will be used in future work for radar performance analysis. We have also discussed real-time primary radar data compression and the decisions we took when implementing this in C++. It has been shown that real-time data compression can be a very useful tool, not only for disk and memory usage, but also to reduce the time spend on reading data for offline analysis afterwards.

References

[1] A. Kruger and W.F. Krajewski, *Efficient Storage Of Weather Radar Data*, Iowa University, Iowa, 1995.

[2] Intersoft Electronics (2009), *Radar Interface Module RIM782*, Available at http://www.intersoft-electronics.com

[3] C. Wolff (2009), *Azimuth Change Pulses*, 16th February 2010 at: http://www.radartutorial.eu/17.bauteile/bt04.en.html

[4] Intersoft Electronics (2009), *Radar Target Generator RTG698*, Available at http://www.intersoft-electronics.com

[5] M. I. Skolnik, *Introduction to radar systems*, 2002, Vol. 3, pp. 49-64.

[6] E. F. Knott, *Radar Cross Section Measurements*, 2004, pp. 14-18.

[7] J.C. Toomay and P. J. Hannen, *Radar Principles for the Non-Specialist*, Vol. 3, 2004, pp. 79-82.

[8] I. Falcounbridge, *Radar fundamentals*, 2002, ch. 14.

[9] M. I. Skolnik, *Introduction to radar systems*, 2002, Vol. 3, ch. 7.

[10] Maxim Integrated Products (2000), *Application Note 641: ADC and DAC glossary*, Available at http://www.maxim-ic.com

[11] K. Hughes and T. Hughes, *Parallel and distributed programming using C++*, 2004, ch. 4.

Migration of a time-tracking software application (ActiTime)

Maarten Devos[1], Ward Vleegen[2], Tom Croonenborghs[1]
[1]IBW, K.H. Kempen (Associatie KULeuven), B-2440 Geel, Belgium
[2]IT responsible, Flanders' DRIVE, B-3920 Lommel, Belgium
Email: devosmaarten@hotmail.com, ward.vleegen@flandersdrive.be, tom.croonenborghs@edu.khk.be

***Abstract*—When the concept of time tracking was introduced for the first time it was used to simply determine the payroll of an employee. The amount of time that was spent on a task could be converted to a reasonable payment for an employee. More useful time spent on a company task, translated itself into a higher payment. These days, time tracking has evolved to a great and handful tool to derivate several important things like how much time is spent on a project, how an employee divides its time onto several tasks etc. Time tracking can determine customer billing information by calculating how much time was spent on a customer project. Flanders' DRIVE uses a free software tool to track time of several employees[1]. This software tool is called ActiTime. The ActiTime application is a free application to register time dedicated to specific tasks. Flanders' DRIVE decided to introduce a new IT infrastructure to meet its business requirements. With the migration from the old infrastructure to the new one, ActiTime also needed to be migrated. A few problems came up in the migration process such as e.g. how to convert the current database, which web server application would be best to use, which server is best suited to install the application etc. In the migration process of the ActiTime application, hyper-V is used to set up the new environment and a little problem with the antivirus real time scan came up. Step by step different problems were solved with a successful migration of ActiTime as a result.**

I. Introduction

Flanders' DRIVE is the Flemish competence pool for the vehicle industry. The company was founded in 1996. When Flanders' DRIVE moved to Lommel in 2004, they decided to buy an IT infrastructure that met the requirements at the new office in Lommel. At the end of the year 2008 Flanders' DRIVE decided to renew their IT infrastructure. To renew an IT infrastructure it is important to correctly transfer all the components of the old infrastructure to the new one. The whole transfer of the IT infrastructure and the implementation of new components can be found in my master thesis "Analyse van een nieuwe IT infrastructuur". When an infrastructure has to be migrated, software with specific user data had to be transferred too. This papers handles the migration process of one of these software applications. This software application is a time tracking software tool called ActiTime.

ActiTime is an important tool to track time of employees. Flanders' DRIVE is using this software tool to create a view on how much time is spent on a customer task or a customer project which involves several employees. The client billing information is partially determined from this software tool. Employees who are using this software tool register their time information through a web interface because the ActiTime application is a web based tool. As Flanders' DRIVE has the need to introduce a new IT infrastructure, several software applications must be migrated to the new infrastructure. ActiTime is one of them. Several problems appear in the migration process. A proper way of how to extract the current user data from the ActiTime database must be found. ActiTime uses java servlets through a web based application. Since the internet information service (IIS) of Windows server 2008 doesn't support java servlets, a different web server must be chosen. This web server need to support the use of java servlets. The developers of the ActiTime application recommend the use of an Apache Tomcat Server[2]. Since Tomcat is a product of Apache, a few problems must be solved to get this server work with Windows Server 2008. A determination of which server is best to use to install the Apache Tomcat server and get ActiTime to work must be made. There is no server that can be used and a decision is made to create a virtual machine with Microsoft Hyper-V.

II. Migration of ActiTime

A. *Flowchart of migration way*

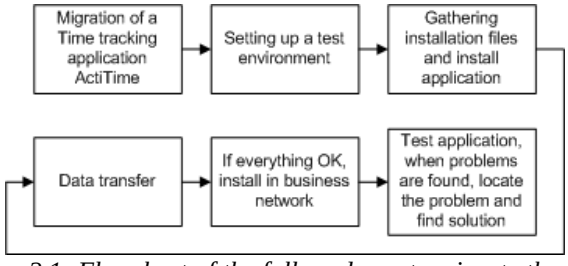

Migration of a Time tracking application ActiTime
Setting up a test environment
Gathering installation files and install application
Data transfer
If everything OK, install in business network
Test application, when problems are found, locate the problem and find solution

Figure 2.1: Flowchart of the followed way to migrate the ActiTime application.

B. Analysis of currently used version of ActiTime and data backup

Flanders' DRIVE is using the ActiTime 1.5 version, installed with an automatic setup. In order to collect the data from the old version, a way to extract the specific user data from the database must be found. It is important to migrate this user data because otherwise, all the time tracking information that was entered before would be lost. The automatic setup allows the administrator to specify which database to use for the collection of the user data. The ActiTime application can run with two database programs: mysql and Microsoft access. When ActiTime was installed for the first time Flanders' DRIVE chose the mysql option. So to extract the data from the old application, a proper way to extract this data must be found. The name of the database could be derived from the ActiTime support files. The database is called 'ActiTime'. The exportation from the user data is a kind of backup that is made. To back up the database the mysqldump[3] command could be used:

mysqldump –u <username> -p<password> ActiTime > actiTime_data.sql

A short explanation of what to fill in:

- <username>: fill out the username that is used to set up the mysql database.
- <password>: fill out the password used for the user who created the database ActiTime. Note that there is no space between p and <password>!
- The parameter after '>' is free to choose. The database backup is stored in the file specified after the '>' symbol.
- The parameter before the '>' sign specifies the name of the used database.

This command can be executed with the Windows command prompt. In the Windows command prompt window you have to navigate to the right directory where the database is stored. Now simply execute the command explained above and a database backup of the ActiTime application is made and saved in the 'actitime_data.sql' file.

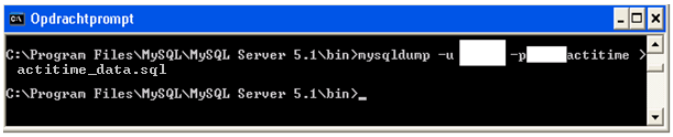

Opdrachtprompt
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysqldump -u -p actitime > actitime_data.sql
C:\Program Files\MySQL\MySQL Server 5.1\bin>

Figure 2.2: command prompt example to extract the user data from the ActiTime database.

C. Setting up a test environment

The installation files of the ActiTime application can be found on the website of ActiTime[4]. In this situation we choose to download the custom installation package. The reason that we choose to download the custom installation package is that in this package customizations in the application can be made. One of the customizations is the java application. With the custom package you can choose which java application to install and which web server to use. For the web server, the Apache Tomcat Server version 6.0.20 is best to use because this web server supports java servlets, the installation of this web server is very straightforward[5]. For the java application, a java runtime engine 6 machine was chosen. ActiTime also need a database to store all the user data. There are 2 options to use, you can choose between MS Access 3.0 or later and MySQL 4.1.17+, 5.0.x+ or 5.1.x+. In this case we choose the MySQL server 5.1 machine. We choose the MySQL option because for this application it suits better than Microsoft Access. However these 2 database system are completely different we still can conclude that MySQL is better in this scenario. Microsoft Access can be very slow if more than 3 clients make a read/write connection to the database simultaneously. Microsoft Access is more a desktop application than an application to use for internet applications. MySQL is more efficient and secure in environments with multiple users connection to the database simultaneously. Microsoft Access has a well developed user interface to create database scheme's while MySQL has no user interface, only a command prompt window to access the database scheme[6]. In the situation of ActiTime we don't need a good user interface because the web application processes the data for us. Since the data is already in an MySQL format it is simpler to migrate the data to a new MySQL server because no database redesign is necessary.

The test environment is set up with virtual machines who can be accessed through Microsoft Virtual PC. The test environment consist of a Windows server 2000 machine with an active directory installed to and two Windows server 2008 machines. Full details on setting up the test environment could be found in "Analyse van een nieuwe IT infrastructuur"[7].

D. ActiTime installation on a test machine

To install the ActiTime application you have to place the installation files in the web-application folder of the Apache Tomcat server. The web-application folder is the directory where the files, needed for a website, are stored and can be viewed by anyone who accesses a specific website stored on our Apache Tomcat server. On our test machine, the ActiTime application files are unzipped to the following directory: *Tomcat 6.0/webapps/ActiTime/*. The application however isn't ready to use yet. To prepare the application to run correctly, a few variables need to be set. These variables specify which database to use, such as the location of the database, the username and password to access the database. To specify these variables for the application, a visual basic script is included in the web folder (*setup_mysql.vbs*). The variables are set and the migration of the old database data to the new database on the new server can start. To insert data into a database, a text file that contains SQL commands can be send to the database. The following command can be used to send the sql file to the database:

mysql –u <username> -p<password> -P <port number> ActiTime < actitime_data.sql

A short explanation what to fill in:

- <username> & <password>: see previous section
- <port number>: The port number that is used to access the SQL database
- The variable before the '<' sign specifies wich database is used

- The variable after the '<' sign specifies the database file, in our case this is the file we've created in the previous section.

To execute this command, the Windows command prompt is used.

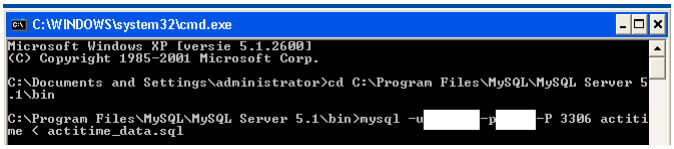

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [versie 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\administrator>cd C:\Program Files\MySQL\MySQL Server 5
.1\bin
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql -u -p -P 3306 actiti
me < actitime_data.sql

Figure 2.3: command prompt example of how to insert the userdata into the new ActiTime database.

The last step is to restart the Tomcat server. When the Tomcat server is reset the ActiTime application can be used. A test to see if the ActiTime application is working like before is made and we check to see if no data is lost. All tests turn out positive and the installation of the application in the business network can start.

E. ActiTime installation on the business network

The next step is to implement the ActiTime application in the operational network. A simple Windows xp machine is used to test the application in the network. After installation of the application and testing it, the application works well and is accessible by the company members. Thereafter, a decision is made to install a new Windows xp machine on a company server. This new Windows xp machine is a virtual computer that is set up through hyper-V (standard component of the Windows server 2008 product). The application is installed in the same way it's described in previous sections. While testing the application, not everything is working as expected. When the Tomcat Server is started, the Tomcat application goes down immediately. This is an unexpected behavior of the Tomcat server application. With this negative behavior of the Tomcat application the ActiTime application is unable to start. A search to a solution for this negative behavior can start.

III. WHY THE TOMCAT APPLICATION WENT DOWN

To solve the problem, first the following different steps are tried to determine the exact problem, but they doesn't lead to a solution of the problem:

- Reinstallation of the ActiTime application
- Reinstallation of the Apache Tomcat Server
- Install a different version of the Apache Tomcat Server
- Reinstallation of the java server machine
- Install a different version of the java server machine

These steps are used to possibly detect the problem, however the problem still exists after each step. After some search on the internet with the terms "can't start tomcat on windows", a possible solution to the problem is found. The solution is tried and indeed, the Apache Tomcat server starts to work again. The problem is a combination of the Apache Tomcat server and the installation of the Java Virtual Machine. The advantage of a Tomcat server over a Windows IIS server is that a Tomcat server can run java servlets. To run these java servlets, Apache has the need to access the Java Virtual Machine that is installed on the machine. When the Apache Tomcat server starts, he searches where to find the java directory and he searches for a specific file "*msvcr71.dll*" in this java directory. This dll file isn't placed in the correct directory when the Java Virtual Machine is installed. To solve this problem we simply copy this dll file in the bin directory of the Tomcat server[8]. The Tomcat application now can find the right dll and starts successfully. The ActiTime application works properly.

IV. HYPER-V

A. What is Hyper-V?

Hyper-V is a role of the Microsoft Windows server 2008 product[9]. With this role, virtual machines can be created and managed. A virtual machine is a simulated computer inside an existing operating system. This operating system runs on its own set of physical hardware. An illustration of how an virtual computer works can be found in figure 4.1 and 4.2.

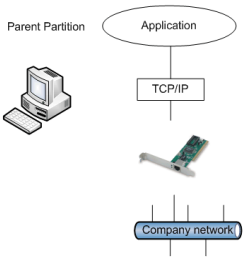


Figure 4.1: Scheme of a normal computer

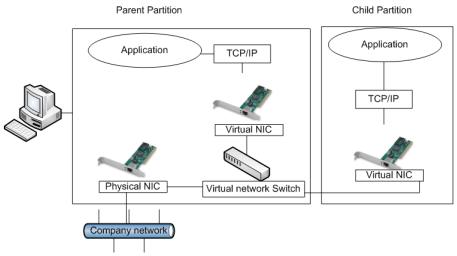


Figure 4.2: Scheme of a virtualized computer

B. Installation of the Hyper-V terminal

Installation of the Hyper-V terminal on the Windows server 2008 product is very straightforward. The installation of the hyper-V terminal can be found in the roles section of the Windows server 2008 product[10]. First open server manager and click on the roles option. Click on the add roles link and an installation wizard is shown. Mark the hyper-V role and click next. An illustration of where to find this role is given in figure 4.3.

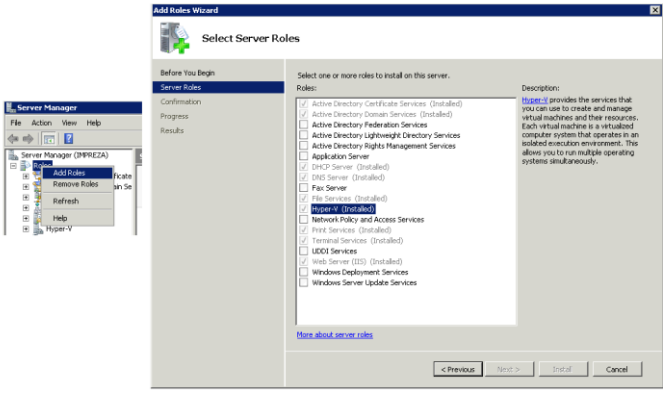


Figure 4.3: installation of the hyper-V role.

Next you can specify the virtual machine specifications[11]. The specifications are not fully listed in this paper. It is important to note that you choose the virtual network adapter by the network preferences. A network adapter is highly recommended because we want full network access for our employees to reach the server through a web browser (intranet). With the virtual network adapter it is possible to register the virtual machine in the business network. With the use of a virtual network adapter the virtual machine act as a real machine that is connected to the business network.

C. *Conflicts between Hyper-V and Trend micro*

When the new virtual machine is created and we turn the machine on, a little problem comes up. The machine turns itself off after a while with an unknown reason. After some search was done, a possible solution for this behavior can be found. The explanation of the problem can be found by the Trend Micro real time scan, which is in use in the whole company. Trend Micro is configured to scan the whole hard disk of the Windows server machine. The directory of the virtual hard disk (file needed for hyper-V where our virtual OS is stored) is scanned by Trend Micro's real time scanning. Since this directory is scanned by trend micro, the vhd (virtual hard disk) file is also scanned. When the vhd file is scanned, hyper-V prevents us to create or start new virtual machines[12]. Hyper-V stops all the virtual machines that are created and are scanned through the trend micro real time scan application. It even lets our virtual machines disappear from the virtual machines list. We find one solution to the problem, untill now it is the only solution available but this solution works. The solution is to add the directory of the virtual machines that are created, to an exclude list of the trend micro real time scan application. You can now say that there is no virus protection on our virtual machine, but there is a workaround. The proper way to protect the virtual machine is to exclude the virtual hard disk directory from the scanning list in the Trend Micro real time scan application and to install the Trend Micro real time scan on the virtual OS. With these modifications the virtual machine starts to run, the ActiTime installation can start and thereafter the virtual machine is known in the company as the ActiTime server.

D. *Why hyper-V*

The ActiTime application contains several components such as the Apache Tomcat web server and the Java Virtual Machine. These components can disrupt other processes or components installed on a Windows server machine. Therefore there must be a proper selection of the right servers we can possibly use to accomplish the role of the ActiTime application. In case of Flanders' DRIVE, the new infrastructure exist of several servers to choose to install the ActiTime application. However no server is found to install the role to. There are no specific rules to determine which server is best to choose to install an application like ActiTime, but several components can be studied. We have an Microsoft Exchange server, it isn't recommended to install the application on this server. Because this server already has a high load and there is a web server installed to access employee mailboxes through a web interface. We use the Apache Tomcat Server and this server can disrupt the Microsoft IIS server installed on this machine. Another possibility is a new server where Active Directory, DNS, DHCP, the citrix licensing, the backup exec etc. is running. Because we prefer to keep the Active Directory server separated from roles who need to set up a web server, this server isn't the best option. There is also a server where the Citrix remote access application is installed. We don't choose this machine because citrix is also using the IIS web server to connect the citrix application with the internet. It's not a good solution to install two web servers of two different vendors on the same machine. Then there is another option to install ActiTime on the Microsoft SharePoint server. Since this server is also using the Microsoft IIS server (SharePoint is a web based environment) we can't install ActiTime on this server either. Our last option is to virtualize a computer where we can install our application to. We find out that the Active Directory server is the server who is carrying the less load. So a virtualization program can be installed on this server. The choice for a virtual machine is the best option because buying a new server would cost the company additional money to simply run the ActiTime application. There are a lot of virtualization solutions. [13]A few programs that accomplish the task to create and manage virtual machines are: vmware, xen, virtual box, Hyper-V etc. Xen and virtual box are both open source programs and vmware is a program you have to pay for. There is not much of a difference between the different virtualizations programs. Since there is little difference between the applications, we opt for Microsoft hyper-v. We choose the hyper-v solution because of its ease of use and because hyper-v is already included in the Microsoft Windows server product. Just add the role of the hyper-v application and a virtual machine program is up and running.

V. CONCLUSION

In this paper we discussed a way to migrate an application. Because a lot of applications needed to be moved to a new server as explained in the short situation scheme, the steps described in this paper are not the same for every application that has to be migrated. This paper treats a few problems that could possibly come up during the migration process. It is not

likely that for other applications the same problems come up. With a short explanation what time tracking contains, the migration of a software tool for time tracking is treated in this paper. The process of migrating an application and its user data is in most cases not very difficult. However when something goes wrong in the migration process it is often hard to determine the exact problem and to find a solution for it. In the migration process described in this paper we found a problem with the Apache Tomcat server. The problem could be fixed by placing a missing dll of the Java Virtual Machine in the right directory of the Tomcat server. A selection of a possible server to move the application to had to be made. After the selection process we came to the conclusion to set up a virtual machine through hyper-v because there was no server available to run the time tracking application. After we set up a virtual machine through hyper-v a rear problem occurred. The created virtual machines couldn't start and began to disappear in the hyper-v management console. This problem occurred because there was a conflict with the trend micro real time scan application. The conflict could be solved by excluding the virtual machine directory from the real time scan list. In the next figure you can find a short summary of the way followed to come to a working migration of the ActiTime application.

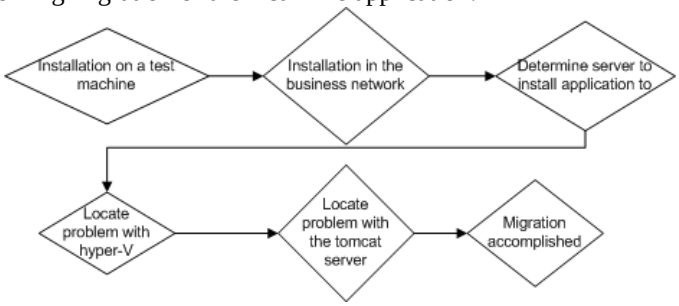

Installation on a test machine
Installation in the business network
Determine server to install application to
Locate problem with hyper-V
Locate problem with the tomcat server
Migration accomplished

Figure 5.1: Summary flowchart of the ActiTime migration

ACKNOWLEDGMENT

I would like to express my special thanks to Flanders' DRIVE who gave me the opportunity to work and learn on their new and old server infrastructure. I also wish to acknowledge Ward Vleegen and Jan Stroobants for their support in my research to the different applications that had to be migrated in the Flanders' DRIVE company and especially the application conducted in this paper, ActiTime. Thanks are also placed for Tom Croonenborghs who coached me through the whole process and gave help and advice to write this paper.

REFERENCES

[1] J. J. Cuadrado-Gallego, *Implementing software measurement programs in non mature small setting.* Software process and product measurement, 2008, pp. 162
[2] http://Tomcat.Apache.org/
[3] V. Vaswani, *Maintenance backup and recovery.* The complete reference of mysql, 2004, pp. 365
[4] http://www.actitime.com/
[5] M. Bond, D. Law, *Installing Jakarta Tomcat.* Tomcat kick start, 2002, pp. 25
[6] M. Kofler, *Microsoft office, openoffice, staroffice.* The definite guide to mysql 5, pp. 120-121
[7] M. Devos, *Onderzoek naar een nieuwe IT infrastructuur*, 2010
[8] Apache Tomcat 6 startup error, available at http://www.iisadmin.co.uk/?p=22
[9] J. Kelbly, M. Sterling, A. Stewart, *Introduction to hyper-V.* Windows server 2008: Insiders' guide to Microsoft's Hypervisor, 2009, pp. 1-4
[10] T. Cerling, J. Buller, C. Enstall, R. Ruiz, *Management.* Mastering Microsoft Virtualization, 2009, pp. 69
[11] A. Velte, J. A. Kappel, T. Velte, *Planning and installation.* Microsoft virtualization with Hyper-V, 2009, pp. 58-59
[12] E-support Trend Micro, available at http://esupport.trendmicro.com/0/Known-issues-in-Worry-Free-Business-Security-(WFBS)-Standard--Advanced-60.aspx
[13] http://nl.wikipedia.org/wiki/Virtualisatie

WAN Optimization Controllers
Riverbed Technology vs. Ipanema Technologies

Nick Goyvaerts[1], Niko Vanzeebroeck[2], Staf Vermeulen[1]

[1]IBW, K.H. Kempen (Associatie KULeuven), Kleinhoefstraat 4, B-2440 Geel, Belgium
[2]Telindus nv, Geldenaaksebaan 335, B-3001 Heverlee, Belgium

nick.goyvaerts@telenet.be
niko.vanzeebroeck@telindus.be
staf.vermeulen@khk.be

***Abstract*—WAN Optimization Controllers (WOCs) become more and more important for enterprises because of the IT centralization. Telindus offers WOC solutions from Riverbed to their customers and Belgacom offers WOC solutions from Ipanema to their customers. Because Telindus now belongs to Belgacom, it is useful to know which the appropriate solution is for a certain customer or network. Riverbed uses the Riverbed Optimization System (RiOS) to optimize WAN traffic. RiOS consists of four main parts, namely data streamlining, transport streamlining, application streamlining and management streamlining. Ipanema uses the Autonomic Networking System or Ipanema system to optimize WAN traffic. The Ipanema system is a managed system that consists of three main parts, namely intelligent visibility, intelligent optimization and intelligent acceleration. Both WOC solutions have similar features but Riverbed has some additional features that Ipanema doesn't have. This paper describes and compares both WOC solutions.**

I. Introduction and Related Work

A WOC is a customer premises equipment (CPE) that is typically connected to the LAN side of WAN routers. These devices are deployed symmetrically on either end of a WAN link (in data centers and remote locations) to improve the application response times. The WOC technologies use protocol optimization techniques to prevent network latency. They also use compression or caching to reduce data travelling over the WAN and they prioritize traffic streams according to business needs. Therefore WOCs can also help organizations to avoid costly bandwidth upgrades.

Telindus offers WOC solutions from Riverbed Technology to their customers and Belgacom offers WOC solutions from Ipanema Technologies to their customers. Because Telindus now belongs to Belgacom, it is useful to know which the appropriate solution is for a certain customer or network. This vendor selection can be difficult because vendors offer different combinations of features to distinguish themselves. Therefore it is important to understand the applications and services (and their protocols) that are running on the network before choosing a vendor. It is also useful to conduct a detailed analysis of the network traffic to identify specific problems. Finally, it's possible to insist on a Proof of Concept (POC) to see how the WOC performs in the company network before committing to any purchase.

Riverbed Technology delivers WOC capabilities through their Steelhead appliances and the Steelhead Mobile client software. It has a leading vision, a great product reputation and some features that Ipanema doesn't have.

Ipanema Technologies delivers WOC capabilities through their IP|engine appliances. It delivers WAN optimization as a managed service.

These WOC solutions are described and compared in the following chapters of this paper.

II. Riverbed Technology

A. Riverbed Optimization System

The Riverbed Optimization System or RiOS is the software that runs on the Steelhead appliances and the Steelhead Mobile client software. RiOS helps organizations to dramatically simplify, accelerate and consolidate their IT infrastructure. RiOS provides the following benefits to enterprises:

- More user productivity,
- Consolidated IT infrastructure,
- Reduced bandwidth utilization,
- Enhanced backup, recovery and replication,
- Improved data security,
- Secure application acceleration.

RiOS consists of four major groups:

- Data Streamlining,
- Transport Streamlining,
- Application Streamlining,
- Management Streamlining.

B. Data Streamlining

Data streamlining or Scalable Data Referencing (SDR) can reduce the WAN bandwidth utilization by 60 to 95 % and it can eliminate redundant data transfers at the byte-sequence level. Therefore even small changes of a file, e.g. changing the file name can be detected. Data streamlining works across all TCP-based applications and across all TCP-based protocols. It ensures that the same data is never sent more than once over the WAN.

RiOS intercepts and analyzes TCP traffic. Then it segments and indexes the data. Once the data has been indexed, it is compared to the data on the disk. If the data exists on the disk, a small reference is sent across the WAN instead of the entire data. RiOS uses a hierarchical structure whereby a single reference can represent many segments and thus multiple megabytes of data. This process is also called *data deduplication*.

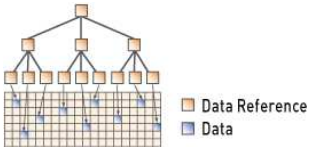


Figure 1 Data references to reduce the amount of data sent across the WAN

If the data doesn't exist on the disk, the segments are compressed using a Lempel-Ziv (LZ) compression algorithm and sent to the Steelhead appliance on the other side of the WAN which also stores the segments of data on disk. Finally, the original traffic is reconstructed using new data and references to existing data and passed through to the client.

C. Transport Streamlining

RiOS uses transport streamlining to overcome the chattiness of transport protocols by reducing the number of round trips. It uses a combination of window scaling, intelligent repacking of payloads, connection management and other protocol optimization techniques.

RiOS uses *window scaling* and *virtual window expansion (VWE)* to increase the number of bytes that can be transmitted without an acknowledgement. When the amount of data per round trip increases, the net throughput increases also. This window expansion is called *virtual* because RiOS repacks TCP payloads with data and data references. A data reference can represent a large amount of data and therefore virtually expand a TCP frame.

The RiOS implementations of High Speed TCP (HS-TCP) and Max Speed TCP (MX-TCP) can accelerate TCP-based applications even when round-trip latencies are high. HS-TCP uses the characteristics and benefits of TCP like safe congestion control. In contrast, MX-TCP is designed to use a predetermined amount of bandwidth regardless of congestion or packet loss.

Connection pooling enables RiOS to maintain a pool of open connections for short-lived TCP connections which reduces the overhead by 50 % or more.

The SSL acceleration capability of RiOS can accelerate SSL-encrypted traffic while keeping all private keys within the data center and without requiring fake certificates in branch offices.

D. Application Streamlining

RiOS is application independent, so it can optimize all applications. There is a possibility to add additional layer 7 acceleration to protocols through transaction prediction and pre-population features.

Transparent pre-population reduces the number of waiting requests that must be transmitted over the WAN. RiOS transmits the segments of a file or e-mail to the next Steelhead before the client has requested this file or e-mail. Therefore a user can access this file or e-mail faster.

Transaction prediction (TP) optimizes the network latency. The Steelhead appliances intercept and compare every transaction with a database that contains all previous transactions. Next, the Steelhead appliances make decisions about the probability of future events. If there is a great likelihood of a future transaction occurring, the Steelhead appliance performs the transaction rather than waiting for the response from the server to propagate back to the client and then back to the server.

RiOS has a CIFS optimization feature that improves windows file sharing and maintains the appropriate file locking. CIFS or Common Internet File System is a public variation of the Server Message Block (SMB) protocol.

E. Management Streamlining

RiOS was designed to simplify deployment and management of Steelhead appliances. There mustn't be made any changes to servers, clients or routers. The management of a Steelhead appliance can be done through Secure Shell (SSH) command line or a HTTP(S) graphical user interface. The management of a complete network of Steelhead appliances can be done through the Central Management Console (CMC). The CMC is an appliance that provides centralized enterprise management, configuration and reporting.

III. IPANEMA TECHNOLOGIES

A. Autonomic Networking System

Ipanema's autonomic networking system or Ipanema system is an integrated application management system that consists of three feature sets:

- Intelligent Visibility,
- Intelligent Optimization,
- Intelligent Acceleration.

It is designed to manage up to very large enterprise WANs. Belgacom offers application performance management (APM) services to their customers through the Explore platform. So the Ipanema system is a managed service.

B. Intelligent Visibility

Intelligent visibility enables full control over the network

and application behavior. It uses IP|engines to gather real-time network information. The IP|engines sent this information to the central software (IP|boss). A synchronized global table stores volume and quality information of all active connections.

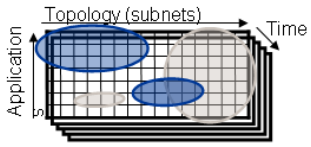


Figure 2 Synchronized global table

The Ipanema system measures application flow quality metrics such as TCP RTT (Round Trip Time), TCP SRT (Server Response Time) and TCP Retransmits. It also uses one-way metrics to measure the performance of a protocol such as UDP (User Datagram Protocol) which is used by VoIP (Voice over IP) and video. Ipanema provides two application quality indicators: MOS (Mean Opinion Score) and AQS (Application Quality Score).

C. *Intelligent Optimization*

Intelligent optimization guarantees the performance of critical applications under all circumstances.

The Ipanema system uses objective-based traffic management to define what resources the network should deliver to each end-user application flow. The enterprises need to define which applications matter the most for them and what the criticalities are for their business. An application with a high criticality is an important application for the business. An application with a lower criticality can tolerate lower quality in a time of high demand. There must also be set a per user service level for each application. This per user service level defines what the network should deliver in terms of network resource for each user of a given application.

IP|engines exchange real-time information about the flows they are controlling. If the cooperating IP|engines detect that they are both sending to the same destination, they dynamically compute the bandwidth for each user session to this destination. This computation or dynamic bandwidth allocation (DBA) is based on their shared knowledge of the traffic mix, its business criticality and the available resources at the destination. The destination doesn't have to be equipped with an appliance to prevent congestions. This is also called cooperative tele-optimization.

Ipanema's smart packet forwarding forwards packets that are belonging to real-time flows. Jitter, delay and packet losses are therefore avoided.

Ipanema's smart path selection dynamically selects the best network path for each session in order to maximize application performance, security and network usage. The network path is calculated using:

- Path resources, quality and availability,
- Application performance SLAs (Service Level Agreements),
- Sensitivity level of the information carried in the flow.

D. *Intelligent Acceleration*

Intelligent acceleration reduces the response time of applications over the WAN so that users get the appropriate Quality of Experience (QoE).

TCP has a slow start mechanism that tries to discover what the available bandwidth is for each session. This mechanism slowly increases the throughput until the link is congested. It assumes then that it has found the maximum available bandwidth. Ipanema's TCP acceleration immediately sets each session to its optimum bandwidth. This leads to the improvement of the response time of many applications, such as those based on HTTP(S). Ipanema can deliver this TCP acceleration without an IP|engine in the branches. Devices are only required at the source of the application flows. This is called tele-acceleration.

Ipanema's multi-level redundancy elimination compresses and locally caches traffic patterns in a cache in the IP|engines of the branch offices. This reduces the amount of data transmitted over the network. Multi level redundancy elimination uses both RAM (Random Access Memory) and disk caches. Therefore it can compress and cache the traffic patterns of very large files and keep them longtime. RAM caches have a smaller compression ratio than disk caches.

Intelligent protocol transformation can optimize protocols to minimize the response time of applications.

IV. COMPARISON BETWEEN BOTH SOLUTIONS

A. *Lab*

We have created an equivalent test lab for both solutions to see which solution performs the best in this simple network environment.

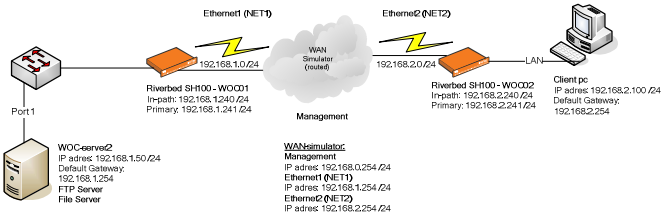


Figure 3 Riverbed Technology lab

File name	FileZilla_3.2.6.1_win32-setup.exe	image_rbt_cmc_2_1_5_n23.img	FileZilla_3.2.6.1_win32-setup_RENAMED.exe	image_rbt_cmc_2_1_5_n23.img (met gewijzigde inhoud)
File size	3,81 MB	39,2 MB	3,81 MB	39,2 MB
Transmission speed/time without WOCs	65,5 KB/s / 61 seconds	78,6 KB/s / 8 minutes 42 seconds	/	/
Transmission speed/time WOCs in bypass mode	65,5 KB/s / 61 seconds	78,6 KB/s / 8 minutes 42 seconds	/	/
Transmission speed/time WOCs with cold optimization	589,8 KB/s / 6 seconds	161,3 KB/s / 4 minutes 14 seconds	/	/
Transmission speed/time WOCs with warm optimization	3,9 MB/s / 1 seconds	12,2 MB/s / 3 seconds	3,8 MB/s / 1 second	10,5 MB/s / 3 seconds

Table 1 Riverbed Technology results FTP-server

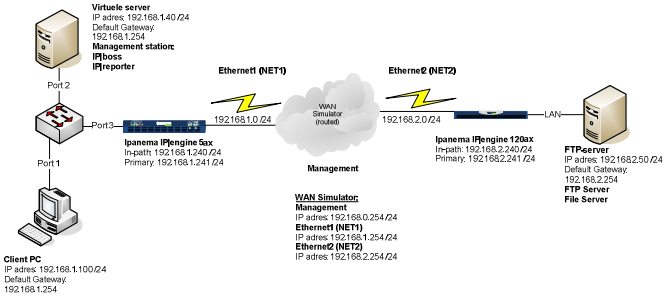


Figure 4 Ipanema Technologies lab

File name	FileZilla_3.2.6.1_win32-setup.exe	image_rbt_cmc_2_1_5_n23.img	FileZilla_3.2.6.1_win32-setup_RENAMED.exe	image_rbt_cmc_2_1_5_n23.img (met gewijzigde inhoud)
File size	3,81 MB	39,2 MB	3,81 MB	39,2 MB
Transmission speed/time without IP|engines	65,5 KB/s / 61 seconds	78,6 KB/s / 8 minutes 42 seconds	/	/
Transmission speed/time IP|engines in failsafe mode	65,5 KB/s / 61 seconds	78,6 KB/s / 8 minutes 42 seconds	/	/
Transmission speed/time IP|engines first time file transfer	120,1 KB/s / 33 seconds	190,0 KB/s / 3 minutes 36 seconds	/	/
Transmission speed/time IP|engines second time file transfer	3,3 MB/s / 1 seconds	6,1 MB/s / 6 seconds	3,2 MB/s / 1 second	3,3 MB/s / 12 seconds

Table 2 Ipanema Technologies results FTP-server

B. Devices

Riverbed Technology uses Steelhead appliances that are placed on both sides of the WAN. There is also a possibility to install the Steelhead Mobile client software on laptops of the mobile users. When the Steelhead Mobile client software is used, there must also be placed a Steelhead Mobile Controller (SMC) in the network. The management of the Steelheads can be done through the management console of the appliance or through the Central Management Console (CMC). The CMC is a device that can manage multiple Steelheads.

Ipanema Technologies uses IP|engines that are placed on both sides of the WAN. There are also virtual IP|engines that must be configured in the management system IP|boss. These virtual IP|engines are especially efficient for very large networks (VLNs).

C. Pricing

Riverbed uses a CAPEX (Capital Expenditures) model. Therefore customers must buy the Steelhead devices.

Ipanema uses an OPEX (Operating Expenditures) model. Belgacom offers Ipanema as a managed service to their customers by which they must pay a monthly fee.

	Physical equipped sites			Physical and virtual equipped sites		
	2 sites network	4 sites network	100 sites network	2 sites network	4 sites network	100 sites network
Riverbed						
With Telindus maintenance	15325,27	52944,46	1003083,51	15325,27	52944,46	936546,89
Without Telindus maintenance	14670,08	50937,49	964300,66	14670,08	50937,49	900095,74
With Telindus maintenance and discounts	13171	45140,73	854724,65	13171	45140,73	798225,51
Without Telindus maintenance and discounts	12515,81	43133,76	815941,80	12515,81	43133,76	761774,36
Ipanema						
Monitoring and optimization	24548	57880	1211056	17674	37258	249730
Monitoring, optimization and redundancy elimination	33188	78040	1582576	23434	/	/

Table 3 Pricing Riverbed and Ipanema for a three year contract in EUR

D. Features

Riverbed Technology		**Ipanema Technologies**	
Data streamlining	Data deduplication → disk-based architecture	Multi-level redundancy elimination → RAM and disk based architecture	
	Encrypted data store	/	
Transport streamlining	Window scaling and virtual window expansion (VWE)	Virtually unlimited TCP window size	
	High speed TCP and Max speed TCP	/	
	Connection pooling	/	
	SSL acceleration	/	
Application streamlining	Transaction Prediction (TP)	≈ Bandwidth tracking	
	Transparent pre-population	Application dependant optimization	
	Windows file sharing (CIFS) optimization	Intelligent protocol transformation/CIFS acceleration	
	SMB signing support	/	
	Lotus Notes optimization	/	
	Exchange (MAPI) optimization	/	
	HTTP(S) acceleration	/	
	Oracle 11i optimization	/	
	Disaster recovery acceleration	/	
	/	Intelligent acceleration	
Management streamlining	WDS device configuration and management	Managed service framework → IP|boss	
	Comprehensive network visibility	Intelligent visibility	
	Central Management Console (CMC)	Managed service framework → IP|boss	
	Steelhead Mobile Controller (SMC)	Managed service framework → IP|boss	
/		Intelligent optimization	Application performance objectives per user
/			Cooperative tele-optimization
/			Smart packet forwarding
/			Smart path selection
/		Network application governance	
/		Rightsizing	

Table 4 Riverbed and Ipanema features

E. Discussion

A file transfer with WOCs placed in the network is faster than a file transfer without WOCs placed in the network. When the appliances are in bypass (failsafe) mode, the transmission time of a file is the same as in a network without appliances. In a network with appliances, the second transmission of a file is faster than the first transmission because the file is stored in memory. When the file is renamed and retransmitted over the WAN, the results are the same as the second transmission of this file. When the content of a file is changed and it is retransmitted over the WAN, the

transmission time increases a little bit because only the changes need to be transmitted unoptimized. From the lab results, we can see that Riverbed optimizes the bandwidth even more than Ipanema. This is especially noticeable with the transmission of larger files.

Both solutions are equivalent when looking at the devices. Riverbed has more features then Ipanema to optimize the network traffic.

When looking at the prices for both solutions, it is obvious that Riverbed is more valuable for physical equipped networks and that Ipanema is more valuable when the network consists of both physical and virtual appliances. This is especially noticeable for networks with many sites. When there are more than five users per site, Riverbed uses a physical appliance rather than a virtual appliance.

V. Conclusion

In this paper we have described and compared two WOC solutions that are offered by Telindus and Belgacom to their customers to optimize WAN traffic. Telindus offers WOC solutions from Riverbed to their customers and Belgacom offers WOC solutions from Ipanema to their customers. Both solutions have similar features but Riverbed has some additional features that Ipanema doesn't have. Riverbed achieves a higher optimization than Ipanema, because it is the market leader of WAN optimization controllers. Riverbed is more valuable for small networks with a few sites which are equipped with physical devices. Ipanema is more valuable for networks with many sites, because it can equip sites with virtual appliances much faster than Riverbed.

Acknowledgment

We would like to express our gratitude to Vincent Istas (Telindus) for his technical support concerning Riverbed. We would also like to express our gratitude to Rudy Fleerakkers (Belgacom) and Bart Gebruers (Ipanema Techonologies) for their technical support concerning Ipanema Technologies.

References

[1] B. Ashmore, "Steelhead Configuration & Tuning", Riverbed Technology
[2] Ipanema Technologies, "Autonomic Networking: Features and Benefits", Ipanema Technologies, 2009
[3] K. Driscoll, "Network Deployment Options & Sizing", Riverbed Technology
[4] K. Driscoll, "Riverbed Steelhead Technology Overview", Riverbed Technology
[5] B. Holmes, "The Riverbed Optimization System (RiOS) 5.5 – A Technical Overview", Riverbed Technology, 2008
[6] Ipanema Technologies, "Intelligent Acceleration: Features and Benefits", Ipanema Technologies, 2009
[7] Ipanema Technologies, "Ipanema System User Manual 5.2", Ipanema Technologies, 2009
[8] Riverbed Technology, "Riverbed Certified Solutions Professional (RCSP) Study Guide", Riverbed Technology, 2008
[9] A. Rolfe, J. Skorupa, S. Real, "Magic Quadrant for WAN Optimization Controllers", Gartner, 30 June 2009, Available at http://mediaproducts.gartner.com/reprints/riverbed/165875.html
[10] Ipanema Technologies, "Smart Path Selection: Combining Multiple Networks Into One", Ipanema Technologies, 8 July 2009
[11] Ipanema Technologies, "Solution Overview: Guarantee Business Application Performance Across The WAN", Ipanema Technologies, 25 May 2009
[12] Riverbed Technology, "MAPI Transparent Pre-Population", Riverbed Technology
[13] Riverbed Technology, "RiOS", Riverbed Technology, 2009, Available at http://www.riverbed.com/products/technology/
[14] A. Bednarz, "What makes a WAN optimization controller?", Network World, 1 August 2008, Available at http://www.networkworld.com/newsletters/accel/2008/0107netop1.html

Line-of-sight calculation for primitive polygon mesh volumes using ray casting for radiation calculation

K. Henrard [1], R. Nijs [2], J. De Boeck [1]

[1]IBW, K.H. Kempen, B-2440 Geel, Belgium
[2]SCK•CEN, B-2400 Mol, Belgium
karel.henrard@edu.khk.be, robby.nijs@sckcen.be, joan.de.boeck@khk.be

***Abstract*—A line-of-sight in this context is a straight line or ray between two fixed points in a rendered 3D world, populated with primitive volumes (ranging from spheres and boxes to clipped, hollow tori). These volumes are used as building blocks to recreate real-world infrastructure, containing one or more radioactive sources. To find the radioactive dose in a fixed point, caused by one of these sources, we construct a ray connecting the point and the source. The intensity of the dose depends on the type and thickness of the materials it crosses. The aim is to find the distances, traveled along the ray through each volume. In essence, this problem is reduced to determining which volumes are intersected and finding the coordinates of these intersections. A solution using ray casting, a variant of ray tracing, is presented, i.e., a method using ray-surface intersection tests. In this case, ray-triangle intersections are used. Because polygon mesh models are only approximations of real surfaces, the intersections deviate from the real-world values. We test the intersection values for each volume type against real-world values and conclude that the accuracy is highly dependant on the accuracy of the model itself.**

I. INTRODUCTION

To understand the importance of this work, it is necessary to introduce the VISIPLAN 3D ALARA planning tool, a computer application used in the field of radiation protection, developed at the SCK•CEN. Radiation protection studies the harmful effects of ionizing radiation such as gamma rays. It aims to protect people and the environment from those effects. An important concept in this field is ALARA, an acronym for "As Low As Reasonably Achievable". ALARA planning means taking measures to reduce the harmful effects, e.g., by using protective shields, by reducing the time spent near radioactive sources and by reducing the radioactivity of the sources as much as reasonably possible. The VISIPLAN 3D ALARA planning tool allows users to simulate real-world situations and evaluate radioactive doses calculated in this simulation.

VISIPLAN provides the tools to create virtual representations of real-world infrastructure, objects, radioactive sources, etc. using primitive volumes. A primitive volume is a mathematically generated polygon mesh model, which means it's a surface approximation rather than an exact representation. This means that only objects with flat surfaces, such as boxes or hexagonal prisms can be modeled in an exact way. Most objects however have some curved surfaces, introducing approximation errors. The resolution of the approximation allows controlling the amount of the error. The higher the resolution, the more polygons (triangles) are used to render the object. A cylinder with a resolution of six will use six side faces, reducing it to a hexagonal prism, while a resolution of 20 produces a much better approximation at the cost of performance. This explanation of surface approximation seems trivial, but it is crucial in this work because it's this triangulated approximation that is used directly in the calculation of intersections. We can't expect to find accurate coordinates of intersections on a cylindrical storage tank if it's modeled with just six side faces.

A simulation consisting of a scene of 3D objects and at least one radioactive source is used to calculate the radiation dose at a specific point in space. The radiation originating from a source may pass through several objects before it reaches its destination, decreasing in intensity. To calculate the attenuation caused by each object, the source model is covered by a random distribution of source points, each having its own ray to the studied point. This is where the line-of-sight calculation enters the picture. It is used to calculate the distances through each material by finding the intersection points on the surfaces of the objects, which in turn are submitted to further nuclear physical calculations to find the dose corresponding to a single source point. It should be noted that the application requires both the geometry and the material (concrete, iron, water,) of each object, as this information is vital in further calculations. The details considering the nuclear physical models fall outside of the scope of this paper.

Once a method for calculating the dose in a single point is developed, it can be used in a number of applications. One application is the creation of a dose map. A dose map is a 2D map that uses colour codes to indicate different intensities. VISIPLAN allows the user to define a rectangular grid of points, with adjustable dimensions and intervals along the width and length of the grid. The line-of-sight calculation introduced earlier is applied to each point of the grid, providing the necessary intensity values. The resulting grid of

values can be converted to a coloured map, much like a computer screen with coloured pixels. This dose map can be used to determine problematic areas – areas with a high radioactive dose – at a glance.

Another interesting application is the definition and calculation of trajectories. When a person is working near radioactive material, he follows a certain path or trajectory through the working area. Using the line-of-sight method to calculate a multitude of points along the defined trajectory and taking the amount of time spent in each location into account, a total dose can determined for the trajectory. This allows the user to evaluate trajectories and try to find the safest route.

II. Broad Phase

Finding intersections between a ray and a triangulated model is generally an expensive operation. Imagine there are 500 primitive volumes in a scene. A simple cylinder at a resolution of 20 consists of 80 triangles, while a hollow torus at the same resolution consists of as many as 1680 triangles. The number of triangles in such a scene quickly adds up. It's unlikely a single ray intersects every volume in a scene. In many cases, no more than a handful of volumes are intersected. Performing expensive operations on each triangle in the scene isn't very efficient. A common approach to this problem is the use of a broad phase and a narrow phase. The broad phase exists of a simple, inexpensive test we can use once per volume, instead of per triangle, to eliminate the volumes that won't be intersected. This is accomplished with bounding volumes. [1] The narrow phase uses a more complex test to find the exact coordinates of the intersection of the ray with a polygon, which is discussed in the next section.

A bounding volume is defined as the smallest possible volume entirely containing the studied object. In addition, the bounding volume must be easily tested against intersections with a ray. Three types of bounding volumes are used often – spheres, AABBs (axis-aligned bounding boxes) and OBBs (oriented bounding boxes). OBBs generally enclose objects more efficiently than the other volumes, but have more expensive intersection tests. A sphere has a lower enclosing efficiency but it also has the cheapest intersection test. [2] In addition, a sphere is easier to describe than an oriented box. For these two reasons, we chose spheres as our bounding volumes.

A bounding sphere is easily described by determining its center point and radius, which can be easily calculated based upon the polygon mesh. [3] Since our primitive volumes are generated from mathematic formulae however, it's easier to find the center and radius analytically. The vertices of a cylinder for example, are generated from a height, a radius and a position vector that serves as the center point of the bottom circle. It is therefore easier to find the center by adding half of the height to the vertical coordinate of the position vector and submitting this new vector to the same rotation matrix. Finding the radius is just a matter of applying Pythagoras to the known radius of the bottom circle and half of the height. Similar techniques can be used for all the other primitives.

A ray is determined by its starting and ending point. Let P_o be the starting point and P_e the ending point. The direction R_d is defined as the normalized vector pointing from P_o to P_e. P(t) is a point along the ray.

$$P(t) = P_o + t \cdot R_d \tag{1}$$

The intersection test is explained in the figure.

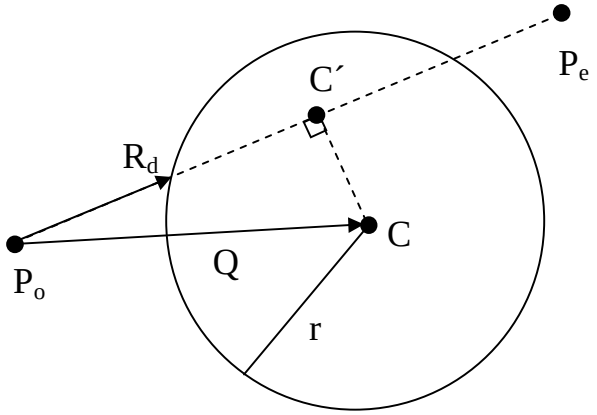


Fig. 1: Intersection of a ray and a sphere

First, vector Q pointing from P_o to the sphere center C is constructed.

$$Q = C - P_o \tag{2}$$

Next, we find the length along the ray between P_o and C' by using the dot product of Q and R_d.

$$\|P_o C'\| = Q \cdot R_d \tag{3}$$

Substituting the t in equation (1) with this length, we can find C' which is the orthogonal projection of the center point C on the ray.

$$C' = P_o + \|P_o C\| \cdot R_d \tag{4}$$

The bounding sphere is intersected if the distance between C and C' is less than the radius r.

$$C = (x_1, y_1, z_1) \quad and \quad C' = (x_2, y_2, z_2)$$

$$d(C, C') = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{5}$$

$$d(C,C') < r \tag{6}$$

One thing we've overlooked so far is that a ray is of infinite length, while we're interested in a ray segment, bounded by the source and the studied point. Imagine the studied point lies between two walls while the source lies outside of these walls. The ray will intersect both walls but the path between the source and the studied point intersects just one wall. In the above test, an intersection is found even if the ray ends before reaching the bounding sphere. To counter this, we'll use an extra test if equation (6) is satisfied.

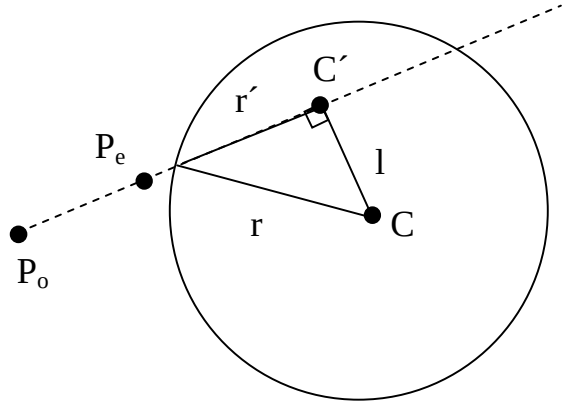


Fig. 2: Halved chord length

$$r' = \sqrt{r^2 - l^2} \tag{7}$$

$$d(P_o, P_e) < d(P_o, C') - r' \tag{8}$$

If equation (8) is satisfied, we can ignore the intersection we found earlier. Note that l is the distance calculated in (5). The effectiveness of the bounding sphere depends on how close the sphere fits the original object. While this certainly is not perfect for long, thin objects, the proposed method provides a considerable increase in performance while inducing reasonable precalculations and programming complexity.

III. Narrow Phase

The broad phase calculations before allow us to eliminate most of the none-intersected volumes from the calculations. The remaining volumes are used in ray-triangle intersections tests. Each volume's triangle list is iterated and each triangle on the list submitted to a test. The test is divided into three stages. In a first stage, the intersection point of the ray with the plane of the triangle is calculated. This requires determining the plane equation, which is a time consuming calculation. Then we check if the intersection is located within (or on) the borders of the triangle. Finally, we'll use another test to check that the ray doesn't end before intersecting the triangle, which is still possible despite the similar test used for the bounding sphere.

A. Plane intersection

Each triangle in the list is defined by three points. Let these points be called P_1, P_2, and P3 and have coordinates:

$$P_1 = (x_1, y_1, z_1)$$
$$P_2 = (x_2, y_2, z_2)$$
$$P_3 = (x_3, y_3, z_3)$$

The plane of the triangle is also defined by these three points, by two vectors between these points or by a single point and the normal vector.

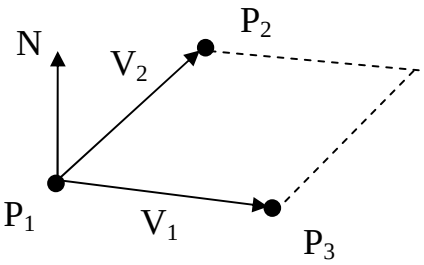


Fig. 3: Plane with three points, two vectors and a normal

$$V_1 = P_3 - P_1 \tag{9}$$

$$V_2 = P_2 - P_1 \tag{10}$$

We find the normal vector by using the cross product.

$$N = V_1 \times V_2 \tag{11}$$

Before we look for an intersection we have to make sure the ray isn't parallel to the plane. That would give us either an infinite amount of intersections or no intersections at all, which are situations we aren't interested in. The condition is:

$$N \cdot R_d \neq 0 \tag{12}$$

An implicit definition of our plane is now:

$$(P(x,y,z) - P_1) \cdot N = 0 \tag{13}$$

Where P(x,y,z) is an arbitrary point. By substituting this point by P(t) from (1), we can find the value of t.

$$t = -\frac{(P_o - P_1) \cdot N}{R_d \cdot N} \tag{14}$$

Using this value in the ray equation (1) returns the intersection point.

B. Point in triangle test

We can check if a point is inside a triangle by using a half-plane test. Each edge of the triangle cuts a plane in half, with one half-plane defined as inside the triangle and the other outside. This test is reduced to three simple equations. [4] P_i is the intersection point.

$$(P_2 - P_1) \times (P_i - P_1) \cdot N >= 0 \quad (15)$$

$$(P_3 - P_2) \times (P_i - P_2) \cdot N >= 0 \quad (16)$$

$$(P_1 - P_3) \times (P_i - P_3) \cdot N >= 0 \quad (17)$$

If all of the above equations are satisfied, the point is inside the triangle. Any equation resulting in a zero means that the intersection is exactly on an edge of the triangle. Such an intersection will be shared by another triangle and could be counted double if the program doesn't take this into account. Other point in polygon strategies exist, but the half-plane test explained above is easily the fastest for triangles. [5]

C. Point between endpoints test

The final test determines whether the intersection is between the starting and ending point of the ray.

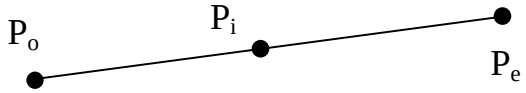


Fig. 4: Point between the endpoints of a line segment

$$d(P_o, P_e) = d(P_o, P_i) + d(P_i, P_e) \quad (18)$$

This equation will only be satisfied if P_i is between P_o and P_e. In any other case, the right hand side will be greater than the left hand side.

IV. Accuracy

The accuracy of the intersections is extremely important for further calculations. The accuracy of the intersections with each type of primitive volume was tested by intersecting them under similar conditions. The idea behind the tests was to analytically calculate the intersections and then compare them against the outcome of the ray-tracer. Each volume was made to intersect with a single ray at different locations of the surface and at different resolutions (20, 50, 100). We let the ray intersect a vertex and the middle of a triangle. The position of a vertex is the exact position of a point on the surface of a volume, while the middle of a triangle is where the model deviates the most from the real surface. The distances in the application are measured in centimeters and we used volumes of different sizes.

Table 1: Deviations of the ray traced intersections at 200 cm, in cm

	Vertex			Triangle		
Resolution	20	50	100	20	50	100
Box	0.000	0.000	0.000	0.000	0.000	0.000
Cylinder	0.000	0.000	0.000	2.191	0.351	0.095
Sphere	0.000	0.000	0.000	2.507	0.368	0.090

In table 1 we show the results for three common volumes of similar sizes – radius, width, depth and height at 200 cm. The tests on the vertices provided perfect results – no errors were measured for these volumes. This means that the method itself is highly accurate; however the problems arise when the intersection is closer to the middle of a triangle. Boxes retain their perfect results when the intersection moves to the middle of the triangle. Curved surfaces however experience significant deviations. At a resolution of 20, a curved volume with a radius of 200 cm can give errors greater than 2 cm. Even at a resolution of 50, there were deviations of a few mm.

Table 2: Deviations of the ray traced intersections at 20 cm, in cm

	Vertex			Triangle		
Resolution	20	50	100	20	50	100
Box	0.000	0.000	0.000	0.000	0.000	0.000
Cylinder	0.000	0.000	0.000	0.214	0.035	0.010
Sphere	0.000	0.000	0.000	0.224	0.036	0.010

In table 2 the same results are shown for volumes with dimension that are 10 times smaller. It seems that the deviations are more or less 10 times smaller as well.

Results vary greatly across the various volumes. Smaller sized volumes naturally have smaller deviations and volumes with a more curved surface generally have greater deviations than those with less curved surfaces. These deviations can't be cured by the method of calculation itself, as they are caused by the difference between a real surface and a polygonal approximation. Increasing the detail of a volume by increasing its resolution provides more accurate results, but this is limited by the hardware specifications.

It is important to note that a previous version of VISIPLAN ensured an accuracy of 0.01 cm, using a different line-of-sight calculation. From the results we conclude that the studied method using ray casting is considerably less accurate for volumes with low resolutions. Only boxes, small sized volumes or volumes with very high resolutions can produce good results.

V. Performance

Another area of interest is the performance of the ray

casting method. While we didn't have access to accurate performance test results of the previous version of VISIPLAN, we know that a line-of-sight calculation to a single point takes about 0.01 second (10 ms) in scene with 30 volumes. In our tests, we used similar scenes of 30 boxes, cylinders or spheres. We also let the number of intersected volumes vary, as this was expected to have a big impact on the performance due to the use of a broad and narrow phase. This is done by simply moving the volumes out of the way so the ray doesn't intersect them anymore, but we'll still have 30 volumes in the scenes.

Table 3: Time required for a line-of-sight calculation, in ms

Intersected volumes	Boxes	Cylinders	Spheres
30	1.63	2.84	16.63
25	1.61	2.71	13.80
20	1.34	2.58	11.35
15	1.17	2.32	9.12
10	1.10	2.19	5.67
5	0.99	2.06	2.97
0	0.91	1.93	0.39

Table 3 shows the time in milliseconds required for a line-of-sight calculation in three different scenes; one with boxes, one with cylinders at a resolution of 20 and one with spheres, again at a resolution of 20. As expected, the time increases significantly as more volumes are intersected; this is especially true for spheres. This can be explained because the polycount – the number of polygons used on the volume – increases more rapidly for spheres when the resolution is increased. We can see that the performance for most scenes is significantly higher than the older method (a few ms as opposed to 10 ms). However in the previous section we concluded a much higher resolution is often needed to reach an acceptable accuracy.

Table 4: Time required for a line-of-sight calculation in ms

Intersected volumes	Cylinders Res 50	Cylinders Res 100	Spheres Res 50	Spheres Res 100
30	5.29	9.54	104.21	417.82
25	5.03	9.41	83.29	348.93
20	4.90	9.28	66.27	279.54
15	4.77	9.15	52.46	207.45
10	4.64	9.02	34.78	138.13
5	4.51	8.90	17.75	69.89
0	4.38	8.64	0.39	0.40

Table 4 shows the results for scenes with cylinders and spheres at higher resolutions. The results look good for cylinders. Even in a scene with cylinders at a high resolution that are all intersected, the time doesn't exceed the 10 ms of the old method. It's a different story for spheres. At higher resolutions the performance deteriorates dramatically. This means that in complicated scenes with many spherical objects, a line-of-sight calculation using the ray casting method may take a lot longer than the old method.

VI. Conclusion

In this paper we showed a method for creating a line-of-sight between two points in a rendered 3D world. Bounding volumes are used as a first, crude filter to reduce the workload. The intersections with polygonal models are then calculated by looking at each triangle of the model. After finding the intersection with the plane of a triangle it is checked whether the intersection is located within the triangle. The test results show that the method itself is accurate, but deviations can be significant if the model isn't detailed enough.

We also conclude that the performance is problematic. A scene consisting of many boxes and other not too complicated volumes can provide the desired accuracy at a very high performance level. More complicated scenes with many spherical objects will struggle either with the accuracy or with the performance of the calculations.

An idea for future work would be to investigate the use of multiple versions of each model at different resolutions, where indices of polygons in a more detailed model could be traced back to indices of polygons in a less detailed model at the same location of the surface. The line-of-sight calculation would start with the least detailed model and work its way up through the more detailed versions, only calculating the polygons near the location of an intersection found in a less detailed model. This method could guarantee a much higher accuracy without the need to calculate an entire model in a high resolution.

VII. References

[1] A. Watt, 3D Computer Graphics, Addison Wesley, 2000, pp. 517-519

[2] A. Watt, *3D Computer Graphics*, Addison Wesley, 2000, pp. 356

[3] "Ray Tracer Specification," Available at http://staff.science.uva.nl/~fontijne/raytracer/files/20020801_rayspec.pdf, February 2010, pp. 5

[4] "CS465 Notes: Simple ray-triangle intersection," Available at http://www.cs.cornell.edu/Courses/cs465/2003fa/homeworks/raytri.pdf, February 2010, pp. 2-5

[5] E. Haines, "Point in Polygon Strategies," in *Graphics Gems IV*, P. Heckbert, Academic Press, 1994, pp. 24-26

Interfacing a solar irradiation sensor with Ethernet based data logger

David Looijmans[1], Jef De Hoon[2], Paul Leroux[1]

[1]IBW, K.H.Kempen (Associatie KULeuven); Kleinhoefstraat 4, B-2440 Geel, Belgium
[2]Porta Capena NV, Kleinhoefstraat 6, B-2440 Geel, Belgium

david.looijmans@portacapena.com
jef.de.hoon@portacapena.com
paul.leroux@khk.be

***Abstract*—In this paper will be described how we interfaced the Carlo Gavazzi CELLSOL 200 Irradiation sensor with the Grin Measurement Agent Control data logger. For this we are required to test the sensor if its output is linear to its input. And also to build and calibrate a microcontroller based circuit to interface the sensor with the data logger. This is required to reach a sample rate of 1Hz or higher to get an accurate energy integral estimate.**

I. Introduction and related work

PORTA Capena is an energy awareness company, that provides a web-based interface Ecoscada. Ecoscada supplies customers with information about their energy and natural resources usage. Locally placed data loggers log sensor and meter data and send it to the Ecoscada database over Ethernet or GPRS. This data can then be accessed through the web-based interface.

With the growing amount of photovoltaic(PV) solar panel installations, there is also an interest in the possibility of confirming if such an installation provided as much electrical energy as it should have done. For this measuring of the solar irradiation is needed.

The system for now makes use of the Grin Measurement Agent Control (MAC), an Ethernet based data logger. The MAC provides 4 Digital outputs, 4 Digital inputs (pulse counters), 4 PT100 inputs, 4 Analog inputs and 1-wire sensors. As well as a 7.5v supply voltage and a calendar function.

The Sensor provided for measuring the solar irradiation is the Carlo Gavazzi Cellsol 200, it's a silicon mono-crystalline cell that works on the same photovoltaic principle as solar panels [4]. The sensor we are provided with is calibrated to give a 78.5mV DC-signal at an irradiation of 1000W/m² and the sensor has a range from 0 to 1500W/m². Because there was no information provided about the linearity of this sensor, the first thing we need to do is test if the output of the sensor is linear with the solar irradiation.

The sensor output is the instant value of the solar irradiation. To reference the sensor output with the electrical energy output of a PV solar panel installation, we are required to integrate the samples over time. For irradiation monitoring a 1Hz sampling rate is recommended minimally to ensure accurate energy integral estimates [1]. However the analog input of the MAC data logger has a maximum sample rate of 1 sample a minute or 0.016Hz. To address this, we plan to setup a microcontroller to sample the sensor output at 1Hz or faster. Then calculate the integral of these values and send pulses on the output accordingly. These can then be logged with the digital input of the MAC data logger.

II. SENSOR LINEARITY RESEARCH

A. *Reference Devices*

For testing the linearity of the Cellsol 200 sensor we require a reference to compare the values. The reference device used was the Avantes AvaSpec-256-USB2 Low Noise Fiber Optic Spectrometer. The specifications of the device can be found in Table1 [2]. And it had a calibration report stating an absolute accuracy of +/-5%.

Wavelength range	200-1100nm
Resolution	0,4-64nm
Stray light	<0,2%
Sensitivity counts/µW per ms integration time	120 (16-bit AD)
Detector	CMOS linear array, 256 pixels
Signal/Noise	2000:1
AD converter	16 bit, 500 kHz
Integration time	0.6 msec – 10 minutes
Interface	USB 2.0 high speed, 480 Mbps RS-232, 115.200 bps

Sample speed with on-board averaging	0,6 msec /scan
Data transfer speed	1,5 ms / scan
Digital IO	HD-26 connector, 2 Analog in, 2 Analog out, 3 Digital in, 12 Digital out, trigger, sync.
Power supply	Default USB power, 350 mA Or with SPU2 external 12VDC, 350 mA
Dimensions, weight	175 x 110 x 44 mm(1 channel), 716 grams

Table1

The spectrometer is connected to a PC by USB2.0 and controlled with the AvaSoft7.4 software that was delivered with the device. It is setup to log the sum of the energy in the wavelength range from 300-1100nm every 30sec. The wavelength range responses to the spectral response of mono-crystalline silicon. The data output is the instantaneous absolute solar irradiation in µW/cm² at a sample rate of 0.033Hz or 1 sample every 30 seconds.

Because the ultimate goal is to compare the sensor output with the energy provided by a PV installation, we will also correlate the sensor data with a PV installation. The PV installation used throughout our research is the setup of the KHKempen. It is made up with 10 Sharp ND-175E1F solar panels with a combined surface of 11.76m² [3]. The panels made of polycrystalline silicon that have an efficiency of up to 12.4%. Other specifications of the panels can be found in Table2.

Maximum Power	(Pmax)	175.0		W
Tolerance		+10	-5	%
Open-Circuit Voltage	(Voc)	29.4		V
Short-Circuit Current	(Isc)	8.10		A
Voltage at Point of Maximum Power	(Vmpp)	23.3		V
Current at Point of Maximum Power	(Impp)	7.52		A
Maximum System Voltage		1000		V
Over-Current Protection		15		A
Application Class		A		
Temperature Coefficient of Pmax		-0.485		%/℃
Temperature Coefficient of Voc		-0.104		V/℃
Temperature Coefficient of Isc		0.053		%/℃

Table2

The converter used is the SMA Sunnyboy 1700 which is equipped with an RS485 interface that allows it to be connected to a PC and allows us to log its input and output. This logs the instant input and output current and voltage, instantaneous absolute output power and the meter reading of the kWh meter every 30 seconds.

For all measurements the Spectrometer and the sensor where installed right next to the solar panel setup, pointing in the same direction under the same angle so that the input for all the 3 setups was the same.

B. CELLSOL 200

For measuring the linearity of sensor an interfacing circuit was needed to transport the sensor signal from the PV installation outside to the data logger inside over a 10m long cable. To prevent the loss of signal strength over the long cable we setup a circuit, at the sensor side, to convert the voltage signal of the sensor to a current signal.

For this we use the AD694 transmitter IC that converts a 0 to 2.5V input to a 0 to 20mA output. Because the sensor needs a high impedance input and to amplify the signal from the sensor to a range of 0 to 2.5V an opamp was used. A second opamp circuit at the data logger side will convert the 0 to 20mA current signal to a 0 to 3V voltage signal.

This resembles the input range for the analog input of the MAC data logger which is 0 to 3V with a precision of 0.01V. This setup was calibrated to give a 3V output voltage for an input voltage of 118mV. 118mV would resemble the maximum output of the sensor at 1500W/m² when we confirm that the sensor is linear. This would also give that precision of 0.01V complies to 5W/m²

C. Results

To determine the linearity of the sensor we need to calculate the correlation coefficient of the correlation between the data from the spectrometer and the sensor. We downsampled the data from the spectrometer the sample rate of the sensor data, being 1 sample per minute. The plot of the 2 signals can be seen in Figure 1.

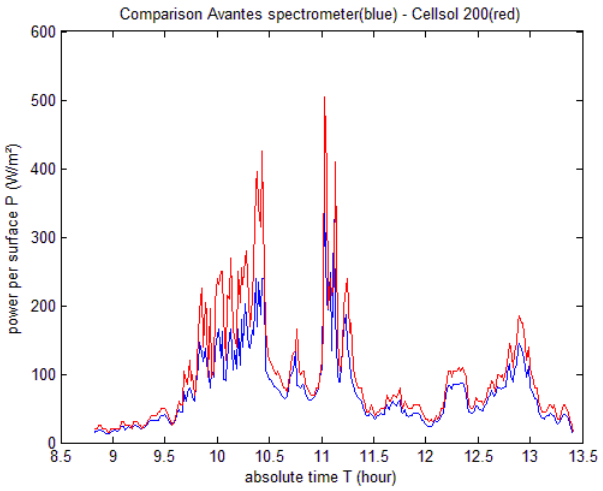


Figure 1

The correlation coefficient between the 2 signals was calculated to be 92.8%, which indicates that there is a large linearity between the 2 signals. However is the sensor signal on average 25% larger than the spectrometer signal. This is probably the result of a calibration error. However this is less important because the calibration for every sensor is different, just as the efficiency of every PV setup will be different. And so they all need to be calibrated after installation.

Secondly we compared the sensor data with the instantaneous absolute power output of the converter. To estimate the correlation between the sensor and the power output of the converter. Figure 2 shows the plot of the signals.

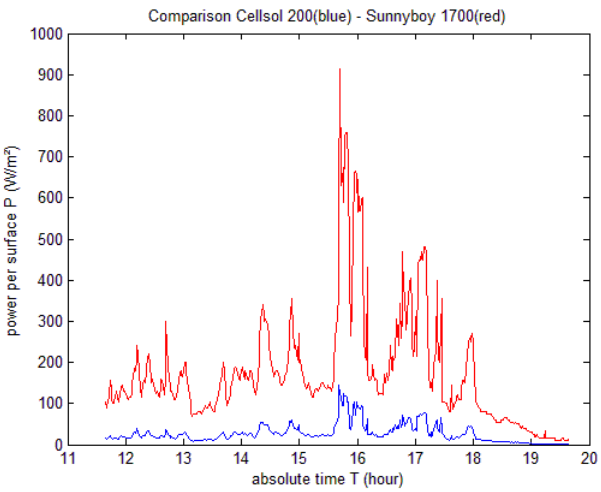


Figure 2

The correlation coefficient between these 2 signals was calculated to be 97.3%. The power output of the PV setup is on average 14.1% of what the sensor indicates. This is explained by the fact that the sensor indicates the power of the incoming solar irradiation and that the Sunnyboy converter the outgoing electrical power. Calculating that the sensor indicates around 25% to much according to the spectrometer this would give an efficiency of 11.3%. This seems acceptable knowing the max efficiency given by the manufacturer of 12.4% and knowing there is still also a loss in the converter.

Out of these results we can deduce that the sensor is linear and that the correlation of the sensor output and the output of the PV setup is high.

III. MICROCONTROLLER CIRCUIT

Now to increase the sample rate and sensitivity of our measurements we introduced a microcontroller based circuit. The intention of this circuit is to sample the sensor output with a much larger sample rate using the ADC of the microcontroller. The microcontroller will add every input value to its buffer. When the buffer value surpasses a predefined threshold value, the buffer will be reset by subtracting the threshold value from the buffer value. When this happens, a digital pulse will be sent at the output.

This resembles integrating the input signal over time, the integration of power over time is energy, so every pulse resembles a measured amount of energy. The pulse output is chosen so that we are able to use the same data logger as it also has a pulse counter. The most common used pulse output in energy meters is the S0 interface described by DIN 43864.

A. The setup

The used microcontroller is the MSP430F2013 from Texas Instruments, it's a chip based on the 16-Bit RISC Architecture that provides us with a 16-Bit Sigma-Delta A/D converter with internal reference and internal amplifier, a 16-Bit timer and several digital outputs [5].

Since there is only 1 timer available we will use this for setting up the sample rate of the ADC as well as the timing for the digital output. So at every clock interrupt the input will be converted and added to the buffer value and then compared with the threshold value. If it exceeds the threshold the output will be set to high. In order to produce a pulse it is required that that next interrupt after the output is set to high, it is always set to low. There for the threshold value must be chosen to be at least 2 times the maximum input. Because the resolution of our setup increases with a lower threshold value, we will set it at exactly 2 times the maximum input. The second parameter that we control that has an influence on the resolution is the sample rate / max pulse rate which is the same for our setup. Devices for the DIN43864 standard require to send pulses of minimum 30ms. This comes down to a sample rate of 33.33Hz.

For the ADC setup we use the internal reference voltage of 1.2V as reference, this gives us an input range from -0.6V to 0.6V. Setting the ADC to unipolar mode and as the max output of the sensor is 117.75mV, we set the internal amplifier to a gain of 4. The resulting input range is 0V to 150mV. The conversion formula for the ADC is:

$$SD16MEM0 = (65536) * \frac{Vin - Vrneg}{Vrpos - Vrneg}$$

With Vrpos = 150mV and Vrneg = 0V. If we insert Vin = 117.75mV in the formula above we get SD16MEM0 = 51446, resulting in a threshold value of 102892. Resulting in that 1 pulse resembles 1500w/m² for 60ms or 0,025Wh/m².

Because the MSP430F2013 does not provide us with a high impedance buffer at the input we are required to implement it ourselves since this is required for the sensor. For this we use an opamp circuit with its gain set to 1.

At the output we use an optocoupler at the output of the circuit that we control with the output of the microcontroller. This is done to limit the current drawn from the microcontroller output and to be able to use larger voltages for the pulse output since the DIN43864 standards gives a voltage range from 0 to 28V.

B. Results

For the measurements the microcontroller circuit was placed on the sensor side and then the pulses transported over the 10m long cable to the data logger inside. This would log every minute the amount of pulses it registered in the past minute. In Figure 3 the energy output of the PV setup is plotted together with the sensor output in an ascending order.

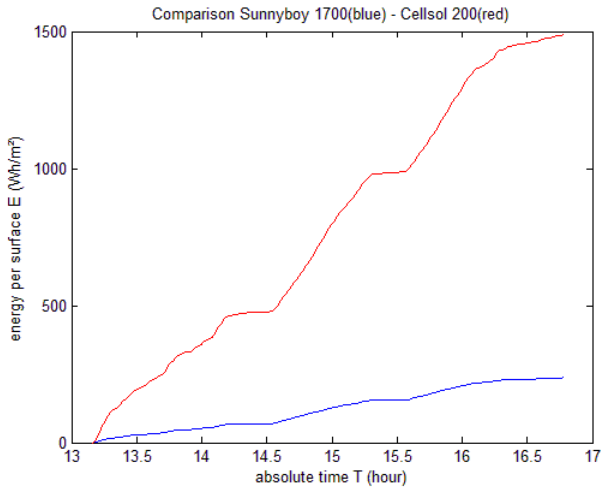


Figure 3

The correlation coefficient between the 2 signals is calculated to be 99.9%, resulting in that the circuit is a good indication for the energy output of the PV setup. The average ratio is calculated to be 15.7%. If we multiply the microcontroller output with this ratio we can see a large resemblance as shown in figure 4.

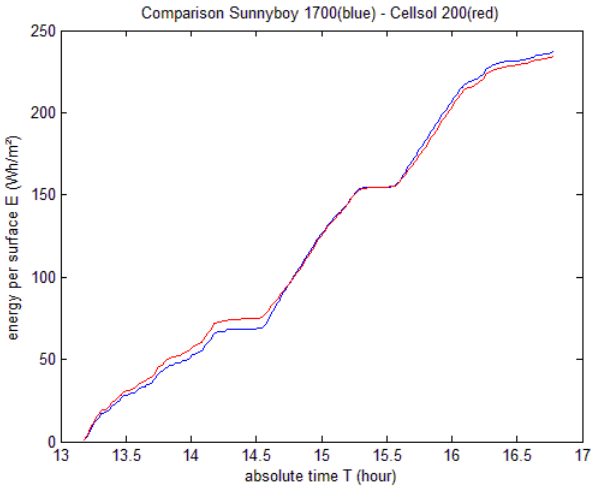


Figure 4

IV. Conclusion

Out of the first part of our research we conclude that the Cellsol 200 sensor is linear and that there is a high correlation with the power output of the PV setup. After implementing the microcontroller circuit together with the Cellsol 200 sensor we become a correlation coefficient of 99.9% between its output data and the energy output of the PV setup. What indicates that this setup is usable to confirm the output of a PV setup.

Acknowledgment

Special thanks go to Wim van Dieren of Imspec for lending us the AvaSpec spectrometer.

References

[1] L. J. B. McArthur, April 2004: Baseline surface radiation network (BSRN/WCRP): Operations Manual. World Climate Research Programme (WMO/ICSU)
[2] Carlo Gavazzi, 2008: Datasheet Irradiation Sensor Model CELLSOL 200
[3] Avantes, April 2009: AvaSpec operating manual
[4] Sharp corporation: Datasheet Solar Module No.ND-175E1F
[5] Texas Instruments August 2005: Datasheet MSP430x20x3 Mixed Signal Microcontroller

Construction and validation of a speech acquisition and signal conditioning system

J. Mertens, P.Karsmakers [1, 2], B. Vanrumste[1]
[1]IBW, K.H. Kempen [Associatie KULeuven],B-2440 Geel, Belgium
[2]ESAT-SCD/SISTA, K.U.Leuven, B-3001 Heverlee, Belgium
[jan.mertens,peter.karsmakers,bart.vanrumste]@khk.be

***Abstract*— In most cases, a close-talk microphone gives an acceptable performance for speech recognition. However this type of microphone is sometimes inconvenient. Other types of microphones such as a PZM, a lavalier microphone, a handheld microphone and a commercial microphone array might offer solutions since these need not be head-mounted. On the other hand due to a larger distance between the speakers mouth and the microphone the recorded speech is more sensitive to reverberation and noise. Suppression techniques are required that increase the speech recognition accuracy to an acceptable level. In this paper, two such noise suppression techniques are explored. First, we have examine the sum and delay beamformer. This beamformer is used to limit the reverberation coming from other angles than the steering angle. Another example is the Generalized Sidelobe Canceller (GSC). The GSC estimates the noise with an adaptive algorithm. Possible implementations of this algorithm are LMS, NLMS and RLS. These 3 types were theoretically as well as practically compared. Speech experiments indicate that compared to the sum and delay beamformer the GSC with LMS gives the best performance for periodic noise.**

***Index Terms*—sum and delay beamformer, Generalized Sidelobe canceller, least square, noise suppression**

I. INTRODUCTION AND RELATED WORK

To change a television station, we can use the remote control by pushing a button. This is the easiest way, but the handicapped aren't able to serve the remote control. In this case voice control will be a viable solution. Here, disabled persons will use their voice i.e. to change the television station. For such systems that use voice control, it's important that the command is recognized by a speech recognizer. For a good recognition, the speech signal has to reach the speech recognizer in a good order. A good microphone placement can solve this problem. This can be achieved with a close-talk microphone. In some situations it is not possible to place a microphone close to the mouth. Thus, we must look to other types of microphones. These microphones will be positioned further away from the speaker. However, we expected problems with reverberation and noise. This results in a decrease in SNR.

In order to increase the SNR there are several techniques:

- Sum and delay beamformer: this beamformer can be used for both dereverberation [1],[2] and noise cancellation [3].
- Adaptive noise cancelling [2]: this is done by LMS.
- Or a combination of the above, e.g. Griffiths Jim Beamformer [2],[3].

In this paper, besides investigating the microphone placement also noise reduction techniques such as mentioned above are examined for periodic and random noise.

This paper is organized as follows. In Section 2 we give an overview of the different microphones in our acquisition system. Section 3 describes the GSC. The sum and delay beamformer and the adaptive algorithms will also be discussed in Section 3 because they are a part of GSC. The results and experiments are reported in Section 4. Finally we conclude in Section 5.

II. ACQUISITION

The goal of the acquisition system is to pick up human speech. This is done with different types of microphones. First, a close-talk microphone is used. This microphone is placed close to the mouth. Due to this small distance noise and reverberation hasn't much influence on the speech. This is an advantage, but the placement of the close-talk microphone can sometimes be annoying. A more comfortable microphone to wear is the lavalier microphone. This microphone is clipped on the clothes. Other microphones which are not attached to the human body are a handheld microphone and PZM. The handheld microphone can be brought close to the mouth, but in this case we have to take

the microphone in hand. This isn't suitable for handicapped persons. So we can place the handheld microphone on a stand, but this might results in a larger distance between speaker and microphone. The PZMs are placed on the four walls of a room.. For the commercial microphone array, we make a similar remark regarding the distance. Finally, every microphone has a polar pattern. This pattern can be omnidirectional, cardioids, hypercardioid or bidirectional. While an omnidirectional pattern records every sound (360°), the other patterns record the sound in a narrower band.

The acquisition system is also composed out of a recorder. This recorder must have the following requirements:

- A sample frequency of 8 kHz or more,
- A resolution of 16 bit or higher,
- Able to record more than 4 channels synchronous,
- Able to record the picked up speech of each microphone on a separate track.

Due to this last requirement we can analyze the data for each microphone individually.

III. Generalized Sidelobe Canceller

The GSC is used to reduce the noise in a speech signal. It consists of 3 parts: a sum and delay beamformer, a blocking matrix and an adaptive algorithm. In figure 1 we see a scheme of the GSC where the inputs y will be the signals picked up by the microphones and the output $\hat{S}_G$ is the enhanced speech signal. Each of the 3 parts is explained next.

A. Sum and Delay beamformer

A beamformer is a system which receives sound waves with a number of microphones. All these sensor signals are processed to a single output signal for achieving a spatial directionality. Due to the directionality, a beamformer can be used for: (i) limiting reverberation [1]; (ii) reducing the noise coming from other directions than the speech. An example of such beamformer is the sum and delay beamformer.

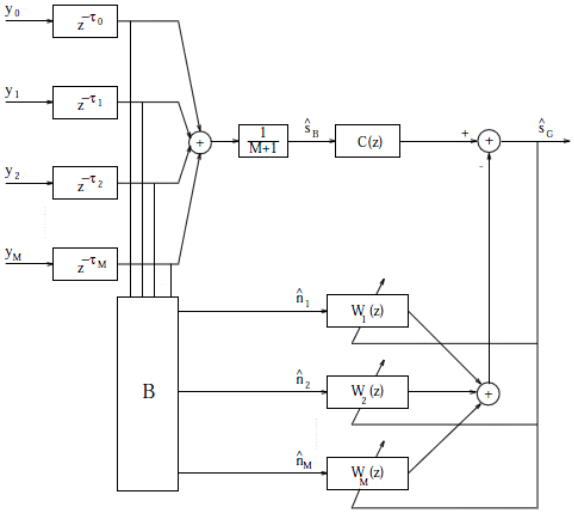


Fig. 1: Generalized Sidelobe Canceller [5]

This beamformer must be steered in the direction of speech. So, a steering angle ϕ is obtained. Figure 2 visualizes this angle.

Because of this steering angle, the microphone signals are delayed against each other. The delay can be calculated using the following manner [9]:

$$\tau = \frac{d \cos \phi}{v} \tag{1}$$

Here, d and v are respectively the distance between two adjacent microphones and the speed of sound ($343 m/s$).

To get the microphone signals in phase, the sum and delay beamformer must add a delay. Expression (1) is used to decide this delay. Afterwards these signals are added together. Finally, the result of the summation is divided through the total numbers of microphones [10].

$$y[k] = \frac{1}{M} . \sum_{m=1}^{M} x_m [k - \Delta_m] \tag{2}$$

Some limitations of the sum and delay beamformer are [2],[3],[4]:

- Limited SNR gain: the SNR slowly increases with the number of microphones.
- Great number of microphones: To obtain a good SNR, we have to use a lot of microphones. This leads to an inefficient array. Non-uniform spacing of the microphones might relax this issue [5].

In the GSC the sum and delay beamformer is useful to obtain a reference signal which is necessary for the adaptive filter in the GSC.

B. Blocking Matrix

The goal of the blocking matrix is to get a reference of the noise at the output. This is obtained by applying a spatial 0 in the steering direction. In this manner the speech is suppressed and we only get the noise.

C. Adaptive filter for SNR-gain

The third part of the GSC is an adaptive filter. The filter is used to estimate the acoustic path of the noise. So at the output of the filter we get an estimation of the noise. The general scheme of an adaptive filter can be seen in figure 3. Here, x[n], y[n] and s[n] are respectively the noise, a filtered version of the noise and the speech.

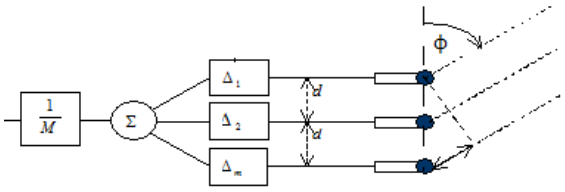


Fig. 2: Sum and delay beamformer with 3 microphones (M=3)

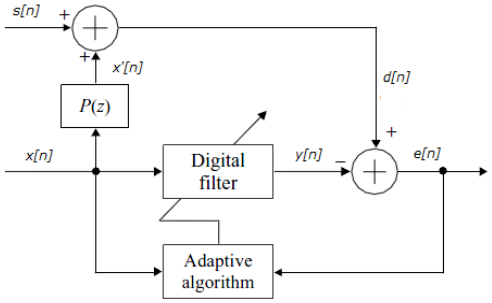


Fig. 3: Adaptive noise cancellation [7]

x'[n] is obtained as x[n] passed the transfer function P(z). Combining x'[n] and s[n] gives the desired signal d[n]. This signal is composed out of speech and noise. The transfer function presents the acoustic path from the noise source to the microphone who records the speech signal. In this manner, it appears that the noise is recorded with the same microphone as the speech signal. Next, the error signal e[n] - calculated by subtracting y[n] from d[n] - is used to adapt the filter coefficients. This adaption can happen on different ways [7][8]. In this paper we discuss 3 algorithms:

- Least Mean Square (LMS)
- Normalized Least Mean Square (NLMS)
- Recursive Least Squares (RLS)

Least Mean Square

LMS tries to minimize the error signal. According to [7] LMS minimizes the following objective

$$w^* = \arg\min_w e^2[n], \tag{3}$$

by adapting the filter coefficients. This boils down to iteratively solving [7]:

$$w[n+1] = w[n] + \mu x[n] e[n] \tag{4}$$

In (4), μ is the convergence factor. This factor controls the stability of the algorithm and also has an influence on the rate of convergence.

The simplicity is the greatest advantage of LMS. This can be seen from (4) where the only operations are an addition and a multiplication.

However, LMS has several disadvantages. If the convergence factor μ is chosen too low. The rate of convergence will be very slow. Increasing μ can solve this problem, but this results in stability problems. Due to a fixed convergence factor, we must find a tradeoff between speed and stability.

Normalized Least Mean Square

This algorithm differs from LMS in the value of the convergence factor μ, which depends on the time. Thus, there is an adaption of μ every time we update the coefficients of the filter. Because of this (4) becomes [7]:

$$w[n+1] = w[n] + \mu[n] x[n] e[n] \tag{5}$$

and μ[n] equals to [7]

$$\mu[n] = \frac{\alpha}{L \hat{P}_x[n]}. \tag{6}$$

In (6), we see three unknown factors. First, we have the factor $\hat{P}_x[n]$. This is the power of x[n] at time n. The power is calculated on a block of L samples. Next, there's a constant α. The value of α lies between 0 and 2. Finally, L represents the filter length.

NLMS solves the problem of LMS. It considers the stability and optimizes the rate of convergence.

A drawback of the algorithm is the extra operation for the calculation of the convergence factor.

Recursive Least Squares

Just like LMS, RLS minimizes the error signal by adapting the filter coefficients. However, RLS uses past error signals for the calculation of the next error signal. The extent to which the previous error signal counts, depends from the forgetting factor λ. This factor is fixed, but the power 'n-i' has as consequence that the older errors have less influence [8]. So the minimization objective is [8]:

$$w^* = \arg\min_w \sum_{i=0}^{n} \lambda^{n-i} e^2[i] \tag{7}$$

This leads to the following iterative formula for determining w[n]:

$$w[n] = w[n-1] + e[n] S_D[n] x[n], \tag{8}$$

where $S_D[n]$ is the autocorrelation of the signal x[n] at time n.

In comparison with LMS, RLS does not depend on the statistics of the signal. Due to this advantage, RLS converges often faster than LMS. However, RLS uses more multiplications [6] per update. This results in a slower algorithm per iteration.

D. Limitations

The blocking matrix in the GSC gives several limitations. These limitations are:

- Reduction of noise in the steer direction: Due to the spatial 0, the noise coming from the same direction as the speech is not suppressed.
- Signal Leakage: Through reverberation, the speech can come from a direction other than the steer direction. In this case the speech will be suppressed. Voice activity detection [10],[11] is required.

IV. Experiments and Results

The goal for the first experiment is to find the most suitable microphone for speech recognition by handicapped persons. For this experiment, we consider two different recording scenarios. The first set of recordings were made in a laboratory setting and have the following characteristics: a reverberant room, ambient noise from a nearby fan of a laptop and test subjects with a normal voice and no functional constraints. The test subjects receive a list with 72 commands which must be spoken out. The recordings were made with a sample frequency of 48 kHz and a resolution of 16 bit. To pick-up the speech, we use different microphones: 4 hypercardioid PZMs at the corners of the room, 1 omnidirectional lavalier, 1 cardioid handheld at a distance of 80 cm, 1 close-talk and a commercial microphone array at 1m in front of the speaker. The setup for the first set of recordings can be seen in figure 4.

The second set of recordings - figure 5 - were made in a real-life setting (the living labs at INHAM) and have the following characteristics: a room with shorter reverberation times, ambient noise from a nearby fan of a laptop and test subjects with functional constraints or pathological voices. In comparison with the setup of the first recording there are 2 differences:

- 4 hypercardioid PZMs are combined to a microphone array with a distance of 0.024 m between 2 adjacent microphones.
- An extra handheld microphone to record the noise source.

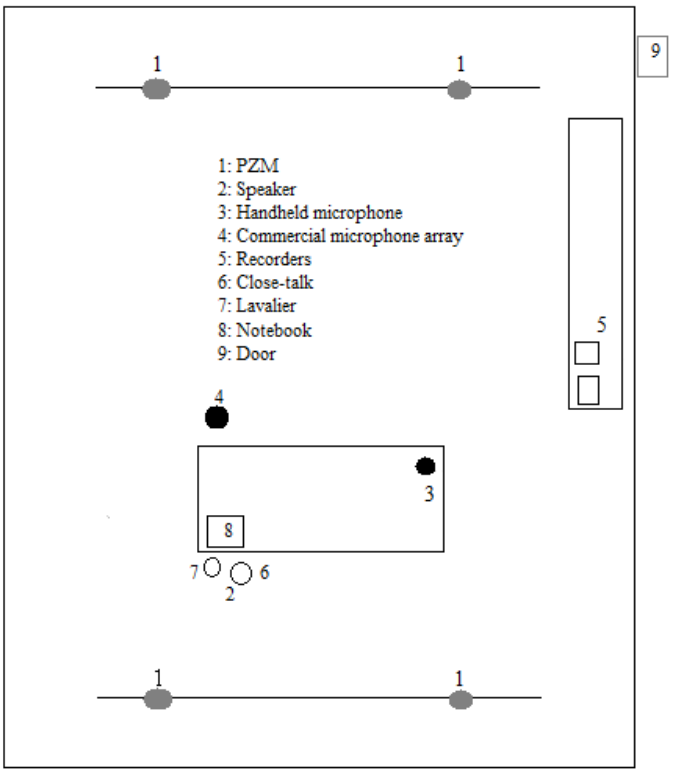


Fig. 4: Setup for the first set of recordings

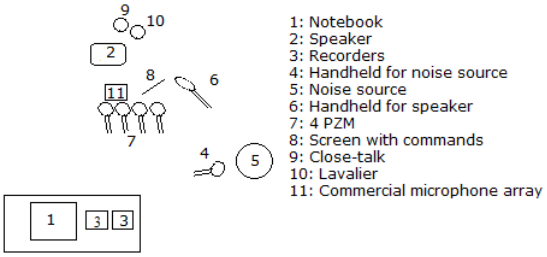


Fig. 5: Setup for the second set of recordings (INHAM)

The recordings were decoded using a state-of-the-art recognition system trained on normal (non pathological) voices recorded with a close-talk microphone. The results of the decoding are given in figure 6 where the Word Error Rate (WER) is defined as [12]

$$WER = \frac{S + D + I}{N_r}, \qquad (9)$$

where S is the number of the replaced words, D the number of substituted words, I the number of inserted words and N_r the total number of words in the reference.

Figure 6 shows that for the first set of recordings the best results were obtained with the close-talk microphone which resulted in a word error rate of 3.6%. Switching to lavalier, the handheld microphone, the PZMs or the commercial microphone array increased the error rate to 4.68%, 16.2%, 30.96% and 43.2% respectively, while the speech recognizer uses state-of-the-art environmental compensation techniques. Based on this results, signal conditioning techniques were required in absence of nearby directional microphone. This is necessary to limit the influence of noise and reverberation. The results for the second set of recordings showed higher error rates. Now, the error rate starts from 48% for a person with a slight speech impairment and going up to 80% and more for pathological voices when using the close-talk microphone. The error rate is also influenced by several factors:

- a short rest in the pronunciation of a command
- dialect of the test subjects
- slower speaking rate
- noise from other persons than the test subject

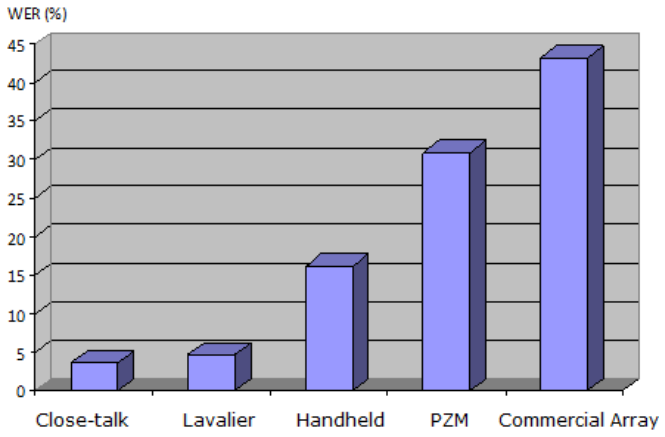


Fig. 6: WER

Based on the results from the first experiment, we investigated some techniques to limit reverberation and noise. For this research, we compare the sum and delay beamformer and the GSC. However, the GSC has an adaptive algorithm. So, we have to examine the most suitable algorithm for this adaptive algorithm. For this experiment, we use the data from the second set of recordings. With figure 3 kept in mind, we combine 10 seconds of data from the close-talk microphone (s[n]) and the handheld microphone for noise (x[n]) to form the desired signal d[n]. The signal d[n] acts, together with x[n] and the corresponding parameters, as input for the 3 algorithms. The parameters are for:

- LMS : convergence factor μ and filter length L
- NLMS: filter length L and constant α
- RLS: filter length L

Afterwards, we calculate the SNR-gain for the different algorithms. The SNR-gain in dB is calculated by taking the difference in SNR between the converged, enhanced signal and the desired signal d[n]. The results for LMS, NLMS and RLS can be found in figure 7,8 and 9 respectively.

We decide to use LMS as adaptive algorithm for the GSC. To obtain the same SNR-gain as LMS with a convergence factor of 0.0050, NLMS has to use larger filter lengths. Next, LMS is much faster per iteration than RLS. Certainly, for the greater filter lengths. Finally, LMS is also much easier in implementation. So taking all these factor into account, we choose for the implementation of LMS as algorithm for the GSC.

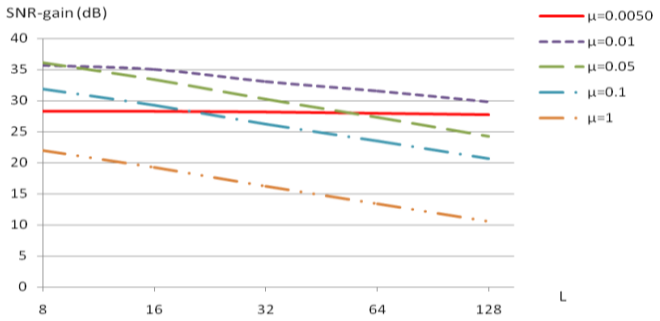


Fig. 7 LMS: influence of the factor μ on the SNR gain

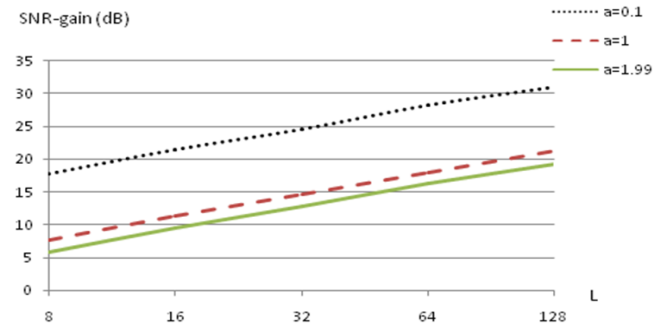


Fig. 8 NLMS: influence of the factor α on the SNR gain

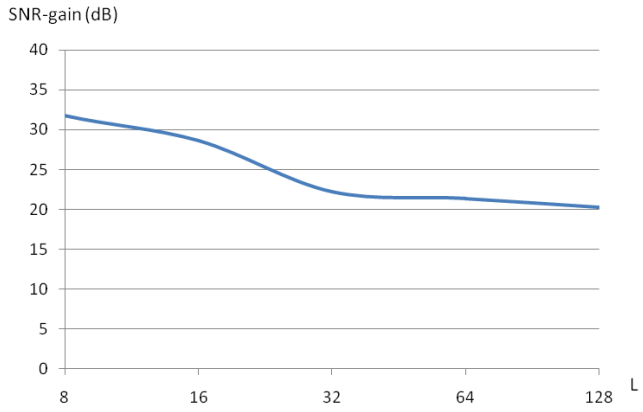


Fig. 9 RLS: SNR-gain

After choosing the adaptive algorithm, the goal of the last experiment is to decide which beamformer (sum and delay beamformer or GSC) is suitable to suppress noise and reverberation and to see what is the effect of adding more microphones and increasing the distance 'd' between 2 microphones in a microphone array. We achieved this by simulating the following microphone arrays:

- Array with 2 hypercardioid PZMs and a distance of 0.024 m between 2 adjacent microphones.
- Array with 4 hypercardioid PZMs and a distance of 0.024 m between 2 adjacent microphones.
- Array with 6 hypercardioid PZMs and a distance of 0.024 m between 2 adjacent microphones.
- Array with 2 hypercardioid PZMs and a distance of 0.072 m between 2 adjacent microphones.

For a microphone array with 2 microphones, we have to generate 2 input signals. To obtain the simulated signals of the microphone array we record a reference signal with the close-talk microphone in the following scenario: reverberant room (veranda with raised curtains), ambient noise from a nearby fan of a laptop, sample frequency of 48 kHz, a 16-bit resolution, test subjects with a normal voice and no functional constraints, speaker in front of the array. Next, we simulate the periodic and/or random noise source at the right side of the array. This is done in MATLAB by adding the corresponding delay to the noise signals. Afterwards, the noise signals must be added to the reference signal to get different desired signals. Now, it is just as if that the simulated signals were caught by the microphone array. Finally, we take from each signal 12 seconds of data – sampled at 8 kHz - as input for the test.

On this data, the SNR-gain is calculated by taking the difference in SNR before and after applying the sum and delay or GSC algorithm. Due to the presence of the adaptive algorithm in the GSC, the GSC algorithm is tested for different convergence factors and filter lengths.

The results for this test can be found in Table 1, Table 2 and Table 3. Here, Table 1 shows the SNR-gain for the different microphone arrays tested on the sum and delay algorithm. Because the sum and delay algorithm is also part of the GSC algorithm an additional SNR-gain is showed in Table 2 and 3. This gain is calculated by subtracting the gain

Table 1: SNR-gain in dB with the use of the sum and delay algorithm in different circumstances: array with 2 microphones and d = 0.024 (A); array with 4 microphones and d = 0.024 m (B); array with 6 microphones and d = 0.024 m (C); array with 2 microphones and d = 0.072 m (D). This table makes also distinction between two types of noise. On one hand periodic noise. On the other hand, random noise.

	A	B	C	D
Periodic	0.21	1.08	2.61	2.04
Random	4.01	6.88	8.75	2.61

Table 2: Additional SNR-gain in dB for the different microphone arrays tested on the GSC algorithm under the presence of periodic noise: array with 2 microphones and d = 0.024 (A); array with 4 microphones and d = 0.024 m (B); array with 6 microphones and d = 0.024 m (C); array with 2 microphones and d = 0.072 m (D). Column L gives the used filter length for LMS with a convergence factor equal to 0.01.

L	A	B	C	D
2	2,32	17,18	8,75	11,48
4	3,21	36,14	15,11	24,01
8	6,41	39,29	28,77	37,49
16	12,76	37,00	36,55	36,82
32	24,68	34,36	34,21	34,26
64	31,41	31,55	31,47	31,50

Table 3: Additional SNR-gain in dB for the different microphone arrays tested on the GSC algorithm under the presence of random noise: array with 2 microphones and d = 0.024 (A); array with 4 microphones and d = 0.024 m(B); array with 6 microphones and d = 0.024 m (C); array with 2 microphones and d = 0.072 m (D). Column L gives the used filter length for LMS with a convergence factor equal to 0.01.

L	A	B	C	D
2	0,01	0,18	0,26	0,01
4	0,02	0,19	0,28	0,01
8	0,02	0,19	0,28	0,01
16	0,02	0,19	0,28	0,01
32	0,02	0,19	0,28	0,01
64	0,01	0,19	0,27	0,01

of the sum and beamformer from the gain of the GSC. Where Table 2 shows the results for periodic noise, Table 3 visualizes the results for random noise.

The last experiment showed that the sum and delay beamformer might offer a good solution to reduce random noise. This can be seen from Table 1 where the SNR-gain for periodic noise is significantly lower than for random noise. However, a GSC doesn't work well with random noise. From Table 3 we see an additional gain of maximum 0.28 dB. This is inferior compared to the results in Table 2. Here, we reach additional gains of 30 dB and more for larger filter lengths. Based on these results we can conclude that a GSC works well with periodic noise. Furthermore, the number of microphones plays also a role for the gain. For the sum and delay beamformer, the results are clear. The SNR-gain increases with the number of microphones. Certainly, for random noise, but this effect can't be seen for the GSC. Moreover, there is no clear dependency between the SNR-gain and the number of microphones. Finally, the distance between 2 microphones is observed. Here, we see no clear relation for the GSC, but periodic and random noise has an influence on the SNR-gain of the sum and delay beamformer. Where the SNR-gain increases for periodic noise, a decrease for the SNR-gain is observed for random noise

V. Conclusion

In this paper we examined the influence of the position of a microphone on the speech recognition. We showed that a microphone near the speaker gives the best performance, but the speaker must have an alternative when there's no possibility to use a close-talk microphone. Due to the greater distance between speaker and microphone all the investigated microphones gave problems with reverberation and noise. So for a good speech recognition this factors must be suppressed. To do this, we applied a sum and delay beamformer and a GSC. A sum and delay beamformer performs better in conditions of random noise, while a GSC with LMS obtains better results in conditions of periodic noise. Finally, increasing the number of microphones gives better results for the reduction of random noise. A better suppression of periodic noise is obtained by increasing the distance between the microphones.

Acknowledgment

The authors want to thank INHAM for their assistance during the recordings which were necessary for this work. In addition we give thanks to ESAT for their investigation with the speech recognizer.

References

[1] K.Eneman, J.Duchateau, M.Moonen, D. Van Compernolle. "Assessment of Dereverberation algorithms for large vocabulary speech recognition systems," Heverlee : KU Leuven – ESAT.

[2] D.Van Compernolle. "DSP techniques in speech enhancement, " Heverlee: KU Leuven – ESAT

[3] D. Van Compernolle, W.Ma, F.Xie and M. Van Diest. "Speech recognition in noisy environments with the aid of microphone arrays," 2nd rev., Heverlee : KU Leuven – ESAT, 28 October 1996.

[4] D.Van Compernolle. Switching adaptive filter for enhancing noisy and reverberant speech from microphone array recordings, Heverlee: KU Leuven – ESAT.

[5] D. Van Compernolle and S.Van Gerven, "Beamforming with microphone arrays*," Heverlee: KU leuven- ESAT* , 1995, pp. 7-14.

[6] B. Van Veen and K. Buckley, "Beamforming : A versatile approach to spatial filtering, " ASSP Magazine, July 1988, pp 17-19.

[7] Kuo, Sen M., *Real-time digital signal processing:implementations and applications* , 2nd ed., Bob H Lee, Wenshun Tian, Chichester: John Wiley & Sons Ltd, 2006, ch.7.

[8] Paulo S.R. Diniz, *Adaptive filtering: algorithms and practical implementation*, 3rd ed., New York: Springer, 2008, ch.5.

[9] I.A. McCowan, *Robust Speech Recognition using Microphone Arrays*, Ph.D. Thesis, Queensland University of Technology, Australia, 2001,pp.15-22

[10] M. Moonen, S.Doclo, *Speech and Audio processing Topic-2: Microphone array processing*, KU Leuven – ESAT.

[11] S. Doclo, *Multi-microphone noise reduction and dereverberation techniques for speech applications*. Ph.D. thesis, 2003.

[12] I. McCowan, D. Moore, J. Dines, D. Flynn, P. Wellner, H. Bourlard, *On the Use of Information Speech Recognition Evaluation*. IDIAP Research Institute, Switzerland, pp.2

Power Management

for

Router Simulation Devices

Jan Smets
Industrial and Biosciences
Katholieke Hogeschool Kempen
GEEL, Belgium

Abstract—Alcatel-Lucent uses relatively cheap Intel based computers to simulate their Service Router operating system. This is a VxWorks based operating system that is mainly used on embedded hardware devices. It has no power management features. Traditional computers have support for power management using the ACPI architecture but need the operating system to manage it. This paper describes how to use the ACPI framework to remotely power off a simulation device. Layer 2 network frames are used to send commands to either the running operating system or powered off simulation device. When powered off, the network interface card cannot receive these frames. Therefore limited power must be restored the PCI bus and network device. Also the network device internal filter must be re-configured to accept network frames that can initiate a wake up. This result is an ACPI compliant system that can be remotely powered off to save energy, and can be powered on when required.

1 Introduction

Alcatel-Lucent's IP Division uses more than 7000 simulation devices. These devices are mostly only used during office hours and left on at night wasting electricity. Some of these run heavy simulations or test suites and must be left on overnight. Every 42-unit rack has a single APC circuit that can be interrupted using a web interface. This will power off all devices within the rack, including the ones with heavy tasks that should have been left on.

The objective is to research and provide the possibility to power off a single simulation device using existing infrastructure and hardware components. If remote power off is possible, it is also required to power on the same device remotely.

2 ACPI

The Advanced Configuration and Power Interface [5] is a specification that provides a common interface for operating system device configuration and power management of both entire systems and devices. The ACPI specification defines a hardware- and software interface with a data structure. This large data structure is populated by the BIOS and can be read by the operating system to configure devices while booting. It contains information about ACPI hardware registers, what I/O address they can be found at and what values there may be written to. The objective is to power off a simulation device. In ACPI terms this maps to the global system state G2/S5, named "Soft Off". No context is saved and a full system boot is required to return to the G0/S0 "Fully Working" system state.

2.1 Hardware Interface

ACPI-compliant hardware implement various registers blocks into the silicon. The Power Management Event Block includes the Status (PM1a_STS) and Enable (PM1a_EN) register. They are both combined to a single event block (PM1a_EVT_BLK). This event block is used for system power state controls, processor power state, power and sleep buttons, etc. If the power button is pressed a bit will raise in the Status register. If the corresponding enable bit is set a Wake Event will be generated.

Another block is the Power Management Control Block (PM1a_CNT_BLK), and can be used to transition to a different sleep state. This block can be used to power off the device.

The General-Purpose Event register block contains an Enable (GPE_EN) register and a Status (GPE_STS) register. These registers are used for all generic features such as Power Management Events (PME). If the corresponding enable bit is set a Wake Event will be generated.

2.2 Software Interface

Each register block is set at a fixed hardware address and cannot be remapped. The silicon manufacturer determines its address location. The ACPI software interfaces

provides a way for the operating system to find out what register blocks are located at what hardware address.

The BIOS populates the ACPI tables and stores the memory location to the Root System Description Pointer (RSDP) into the Extended BIOS Data Area (EBDA). The operating system scans this area for a string "RSD PTR " which is followed by 4 bytes. This 32-bit address is a pointer to the RSDP. At a 16-byte offset the 32-bit address of the Root System Description Table (RSDT) can be found. Figure 1 illustrates this layout.

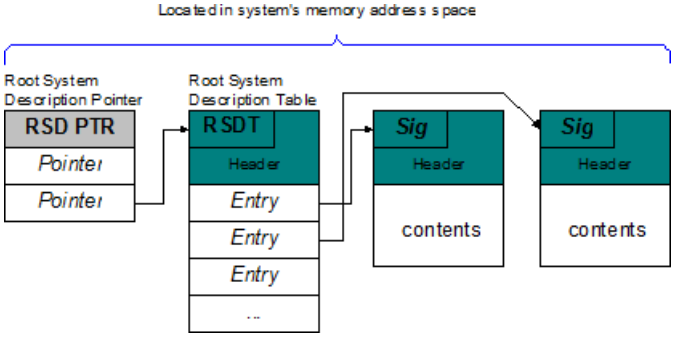


Figure 1. RSD PTR to RSDT layout

From this point on, every table starts with a standard header that contains a signature to identify the table, a checksum for validation and so on. Thus the RSDT table itself contains a standard header, after this header a list of entries can be found. The number of entries can be determined using the length field from the table header.

The first of many RSDT entries is the Fixed ACPI Description Table (FADT). This table is a key element because it contains entries that describe the ACPI features of the hardware. Figure 2 illustrates this.

At different offsets in this table a pointer to the I/O locations of various Power Management registers can be found, for example the PM1a_CNT_BLK. The FADT also contains a pointer to the Differentiated System Description Table (DSDT) table which contains information and descriptions for various system features.

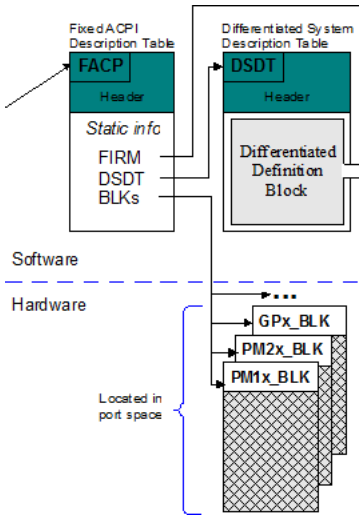


Figure 2. FACP contents.

2.3 PM1_CNT_BLK

This is a 2-byte register and contains two important fields. The SLP_TYP is a three bit wide field that defines the type of hardware sleep the system enters into when enabled. Possible values associated with their sleeping state can be found in the DSDT. When the desired sleeping states is inserted into the SLP_TYP field the hardware must be told to initiate. This is done by writing a one to the one bit field SLP_EN.

2.4 DSDT

The Differentiated System Description Table contains information and descriptions for various system features, mostly vendor specific information of the hardware. For example the DSDT tables contains a S5 object that contains three bits can be written to the The SLP_TYP field.

2.5 Summary

At this point we know what steps need to be taken to power off a simulation device. We can conclude that *it is possible to power off any ACPI compliant system*, which is the case for all motherboards used in simulation devices at Alcatel-Lucent.

3 Remote Control - Power Off

Layer 2 packets are used to send commands to the simulation devices. This means that it can only be used on the same layer 2 domain, e.g. broadcast domain. The packets are captured by the operating system kernel. This means that there is no application on top of the kernel processing incoming packets. This approach is chosen to capture these "management" packets as soon as possible in kernel space so the upper layers cannot be affected in any way. All simulation devices have a unique 6-byte MAC address and a "target name", which is has a maximum length of 32 bytes. Every device uses this target name to identify itself. IP addresses are not unique and may be shared between simulation devices.

3.1 Packet Layout

A layer 2 packet, also known as an Ethernet II frame, starts with a 14-byte MAC header, followed by variable length payload - the data - and ends with a 4-byte checksum.

3.1.1 MAC Header

The MAC header consists of the destination MAC address to identify the target device, followed by the source MAC address, to identify the sending device. At the end of the MAC header there is a 4-byte EtherType field. This identifies the used protocol, for IPv4 it's value is 0x0800. Since we're creating a new protocol, it is suitable to adjust the EtherType field. We have chosen the 2-byte value `0xFFFF` to identify the "management" packets. In this way a possible mix up with other protocols is avoided and the "management" packets complies with IEEE standards.

3.1.2 Payload

Payload is the content of the packet and contains following fields:

- target MAC (6 bytes)
- target name (32 bytes)
- source IP (4 bytes)
- action (1 byte)

The target MAC is also found inside the MAC header, but are not always identical. When using broadcast messages, all devices within that subnet will receive the broadcast packet. In this case it should only be processed by the simulation device it was destined to. The target name is a unique name for every simulation device and is well-suited for identifying the device. Since Layer 2 packets are used, the IP protocol is omitted and no IP addresses are used. The IP source field is included for logging purposes. The action field defines what command the operating system must execute, this gives the possibility to further expand the use of these ”management” packets.

3.2 Processing

All incoming packets are examined by the network interface. All broadcast and unicast packages that match are accepted and passed on. At kernel level all incoming packets are processed. At an early stage, the EtherType of every MAC header is examined to match 0xFFFF. If no match is detected (e.g. other protocol) it is left untouched. If the packet matches, a subroutine is executed and the entire package (MAC header + payload) is passed using pointers. This function further validates the incoming packet and executes the desired command based on the payload's action field.

3.3 Summary

A layer 2 packet layout is designed and can be used execute tasks remotely. One of these task is to initiate a ”Soft Off” command using the information found with the ACPI framework. Combing both the ACPI framework and layer 2 ”management” packets it is possible to remotely power off a router simulation device. *We can hereby conclude that remote power off is possible and can be successfully implemented in an operating system with no power management extensions.*

4 Remote Control - Power On

The last step is to power on the simulation device. When powering off, the entire device is placed into the ACPI G2/S5 ”Soft Off” state. Meaning that all devices are shut down completely. This is a problem since an inactive network device cannot receive network packets or even process them.

4.1 Remote Wake Up

Remote wake up is a technology to wake up a sleeping device, using a special coded ”Magic Packet”. Most network devices support the use of Remote Wake Up, but need auxiliary power to do it. All necessary/minimal power for the network device to receive packets can be provided by the local PCI bus [7]. A second requirement is that the Wake Up Filter is programmed to match ”Magic Packets”. Note that Remote Wake Up is different from Wake On LAN. WOL uses a special signal that runs across a special cable between the network device and motherboard. Remote Wake Up technology uses PCI Power Management [10].

4.1.1 Magic Packet

A Magic Packet is a Layer 2 (Ethernet II) frame [11]. It starts with a classic MAC header that contains destination and source MAC address followed by an EtherType to identify the used protocol. EtherType `0x4208` is used for Magic Packets. The payload starts with 6 bytes `0xFF` followed by sixteen repetitions of the destination MAC address. Sometimes a password is attached at the end of the payload, but not many network devices support this.

4.1.2 Wake Up Registers

Wake up filter configuration is very vendor specific. At Alcatel-Lucent, most simulation devices use an Intel networking device. Wake Up Registers are internal registers that are mapped to PCI I/O space [8].

There are three important Wake Up Registers.

4.1.2.1 WUC: Wake Up Control register. This register contains the Power Management Event Enable bit and is discussed later on at PCI Power Management.

4.1.2.2 WUFC: Wake Up Filter Control register. Bit 1 from this register enables the generation of a Power Management Event upon reception of a Magic Packet.

4.1.2.3 WUS: Wake Up Status register. This register is used to record statistics about all wakeup packets received. Useful for testing.

4.2 PCI Power Management

The PCI Power Management specification [10] provides different power states for PCI busses and PCI functions (devices). Before transitioning to the G2/S5 ”Soft Off” state, the operating system can request auxiliary power for devices that require it. This is done by placing the device itself into a low power state. D3 is the lowest power state, with maximal savings, but enough to provide auxiliary power for the network device.
Every PCI device has a Power Management Register block that contains a Power Management Capabilities (PMC) register and Power Management Control/Status Register (PMCSR). The most important register is the PMCSR. It contains two important fields.

4.2.0.4 PowerState: This field is used to change power state. D3 state provides maximal savings with auxiliary power to provide Remote Wake Up capabilities.

4.2.0.5 PME_En: Enables wake up using Power Management Events. This is the same bit used in the WUC register from the Intel network device.

4.2.1 Wake Event Generation

Wake events can be generated using Power Management Events. The PME signal is connected to pin 19 of a standard PCI connector. Software can assert this signal to generate a PME. That software could be the wake up filter from the Intel network device.

The system still has to decide what to do with the generated PME signal. Recall the ACPI General-Purpose Event register block with corresponding Enable and Status registers. The Status register contains a field named PME_STS that maps to the PME signal used on the Intel network device. All what is left to do is set the corresponding enable bit in the Enable register. When Status and Enable bit are set, a wake event is generated and the system will transition to the G0/S0 "Working" state.

4.3 Summary

When the network device is kept powered on and configured to generate a wake event through a power management event upon reception of a Magic Packet, the system will transition to the "Fully Working" state. *We can conclude that remote power on is possible and can be successfully implemented on simulation devices.*

5 Conclusion

This works shows that it is feasible to implement power management features into the VxWorks operating system that initially had no support for it. Both remote power off and power on are successfully implemented. We can conclude that all goals are achieved.

Acknowledgments

The author would like to express his gratitude to everyone at Alcatel-Lucent IP Division for assisting throughout this work. The author also wants to thank Alain Maes, Erik Neel and Dirk Goethals for their assistance and guidance during implementation of this work. Thanks also go out to Guy Geeraerts for supervising the entire master thesis process. Last but not least, special thanks go out to the author's girlfriend, brother, relatives and friends who encouraged and supported the author during writing of this work.

References

[1] S. Muller, *Upgrading and repairing pcs*, 15th ed. Que/Pearson tech. group, 2004.
[2] Intel Corporation, *Intel 82801EB ICH5 Datasheet* Catalog nr. 252516-001, Available at intel.com, 2003.
[3] Intel Corporation, *Intel ICH9 Datasheet* Catalog nr. 316972-004, Available at intel.com, 2008.
[4] T. Shanley, D. Anderson, *PCI System Architecture* Addison-Wesley Developer's Press, ISBN 0-201-30974-2, 1999.
[5] Hewlett-Packard, Intel, Microsoft, Phoenix, Toshiba , *Advanced Configuration and Power Interface Specification*, ed. 3.0B Available at acpi.info, 2006
[6] Intel Corporation, *Intel 64 and IA-32 Architectures Software Developers Manual*, vol 3B. Catalog nr. 253669-032US, Available at intel.com, 2009.
[7] PCI Special Interest Group, *PCI Local Bus Specification*, rev 2.2 Available at pcisig.com, 1998.
[8] Intel Corporation, *PCIe* GbE Controllers Open Source Software Developers Manual* rev. 1.9 Catalog nr. 316080-010, Available at intel.com, 2008.
[9] Intel Corporation , *ACPI Component Architecture Programmer Reference*, rev. 1.25 Available at acpi.info, 2009
[10] PCI Special Interests Group, *PCI Bus Power Management Interface Specification*, rev 1.2 Available at pcisig.com, 2004.
[11] Lieberman Software Corporation, *White Paper: Wake On Lan*, rev. 2 Available at liebsoft.com, 2006
[12] W. Richards Stevens *TCP/IP Illustrated Vol. 1 - The Protocols* Addison-Wesley, ISBN 0201633469, 2002.

Analyzing and implementation of Monitoring tools (April 2010)

Philip Van den Eynde Kris De Backer Staf Vermeulen
Rescotec
Cipalstraat 3
2440 Geel (Belgium)
Email: Philipvde@pandora.be

***Abstract*—The quest of analyzing monitoring tools that use the least of your network and server capacity, to keep track of all kind of resources (services, events, disk space and BlackBerry Services). One of the objectives that must be met, is the automatic restart of a service when it goes offline. The research starts from here. First of all the tools must be tested in a standard environment where the parameters are always the same. It begins with eliminating the tools that do not have the required objectives, the ten candidate tools are the ones that have it all and will be put in benchmark.**

I. INTRODUCTION

IN large server environments, it is not obvious to manually monitor all running servers and services. For some critical services, it is even unacceptable that they go offline. Therefore, most company networks are automatically monitored by dedicated 'agents', checking the availability of all running services. On the other hand, when networks become large, the additional network overhead caused by these tools cannot be ignored. The research in this paper aims to optimize the downtime of services without using too much of the network bandwidth.

II. DESIGN REQUIREMENTS

A. *Parameters that are necessary in the tool*

The following parameters must be met for a tool, before it is put in benchmark. All the listed items are services or resources that a system admin must check frequently to prevent failures and unwanted downtime.

Some extra information, for people who have no experience with BlackBerry. The “besadmin” is the admin to control BlackBerry services. A list of tools has been checked for the proper specification, for example Nagios [1] did not have the ability to scan with another admin.

Services with local system admin:	Services with Besadmin:
Print Spooler	BlackBerry Alert
Microsoft Exchange Information Store	BlackBerry Attachment Service
Microsoft Exchange Management	BlackBerry Controller
Microsoft Exchange Routing Engine	BlackBerry Dispatcher
Microsoft Exchange System Attendant	BlackBerry MDS Connection Service
Ntbackup (Eventlog)	

Table. 1. Testing parameters

Some examples of tools that didn’t make the benchmark are Internet server monitor, Intellipool, IsItUp, IPhost, Serversalive, Deksi network monitor, Javvin (Easy Network Service Monitor), SCOM, … this because of the limitations or the overall cost.

The tools that fulfill all needs are listed in random order, and will be put in benchmark for comparison:

1. ActiveXperts
2. Ipsentry
3. ManageEngine
4. MonitorMagic
5. PA Server Monitor
6. ServerAssist
7. SolarWinds
8. Spiceworks
9. Tembria server monitor
10. WebWatchBot

B. *Setting up the standard environment*

The environment consists of one small business server, where the services will be running and a monitor server with the appropriate tool for the benchmark. These two servers will be connected with a Cisco 1841 router for a stable network. Both systems run virtually (VM Ware) on two different physical systems with the following specifications.

Fig. 1. Standard testing environment

testserver	monitor server (tool)
Small Business Server 2003	Windows XP Prof SP3
AMD Athlon XP 2500	Intel® Core™2 Duo @ 2.4ghz
384MB RAM	512MB RAM

Table. 2. Standard testing environment

Remark. SolarWinds is a tool that does not follow up the standard environment, because it only runs on a dedicated server environment. Therefore the tool will be installed on a virtual (VM Ware) Small Business Server 2003 instead of the defined Windows XP client.

After setting up the network, the software will be tested on CPU, DISK, memory and network performance. This part is done by Windows Performance monitor [2][3] and WireShark [4] for the network part.

Because it is a small network the statistics that we become will be in a non working network, this results in a lower network load then in real time. Keeping this in mind, we can start the simulations. Later on we will put the best tool for the company in a real time networking environment.

III. SIMULATIONS

The benchmark consists of tests that represent a server environment in real time. Following fields will be tested:

1. A non-successful NTBackup of the “test.txt” file, which will result in an error in the application log file.
2. Full configured Perfomance monitor (onboard Windows testing tool) with the following parameters:
    a. DISK (scale 0-300)
        i. Disk read/sec
        ii. Disk write/sec
        iii. Transfers/sec
    b. CPU (scale 0-100%)
        i. CPU average
    c. RAM
        i. % committed bytes
3. Monitor tool set up with the capability to monitor the previous listed services and events, with a scan frequency of 5 minutes.

During the 30 minutes test process, WireShark will monitor the network load of the specific tool under test.

First of all the tools both run as a service on the monitor server and follow a previous defined procedure therefore we can compare them equally.

time	service that will go down
At start	BlackBerry Dispatcher (Disabled)
4 min	Print Spooler
8 min	MSExchangeSA + MSExchangeIS
15 min	BlackBerry Server Alert
18 min	MSExchangeMGMT
22 min	BlackBerry Controller
25 min	BlackBerry MDS Connection Service

Table. 3. Test procedure

Another specific requirement is the ability to start the service automatically when it goes down, the IT-specialist does not have to intervene.

A. *Benchmarks*

The tools listed before are all tested for the specific 30 minutes testing procedure, because of the large scoop of test results we will limit the results to the summarization of CPU, DISK, memory and network performance.

First of all, our company policy requires the server to run together with other services on a Small Business Server. Our customers do not have the budgets to run such tools on dedicated servers.

This brings us to determine which factor is the most important for the company. We’ve decided that a tool for monitoring purpose to prevent problems, may not cause one by tearing down the network in performance.

The network load of such a tool should not interfere with the normal work of a server room. Followed up by the server load, with as most important factor, the disk operations. As mentioned before, the tool will not run dedicated but together with other servers like SQL Database Servers. Such a server requires all data to be processed and not being lost by scans of a monitoring tool. This means that disk operations, transfers/sec to be precisely, may not reach a certain limit of IO-maps/sec or data can get lost in the process.

Other parameters like memory and CPU are not so important, because servers are powerful machines that most of the time run beneath their capabilities.

Bringing us to the last but not least parameter, the price. Good tools proportionally go with the price. Because the most of our

customers are smaller companies the price should be in the same order.

B. *Network load*

As we take a look at the network load during the 30 minutes scan procedure, it's clear that MonitorMagic has the lowest use of bandwidth.

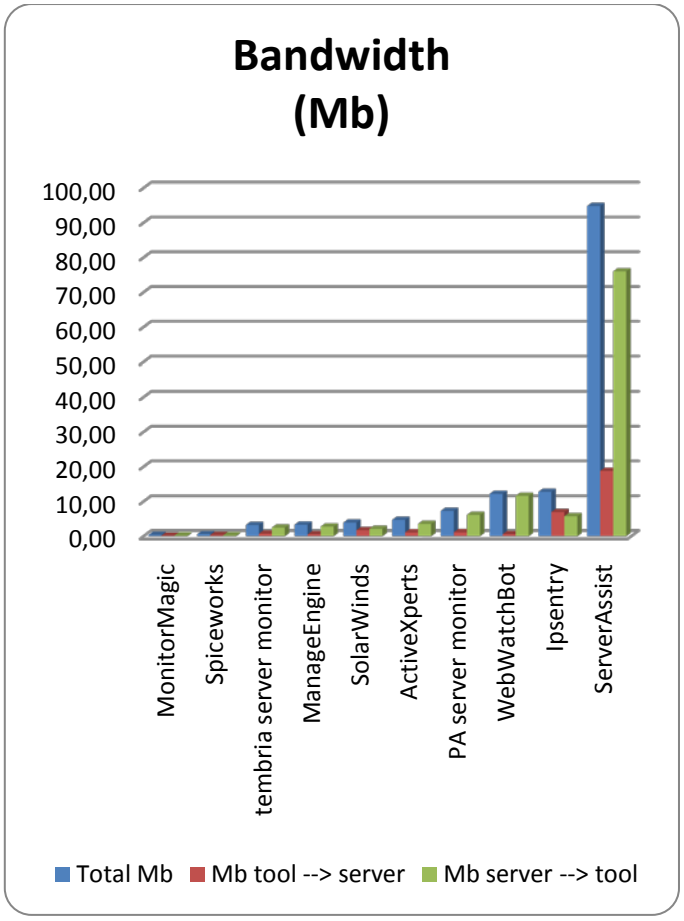


Fig. 2. Bandwidth results

With the details listed in the following table

monitor	Total Mb	tool-- > server	server --> tool
MonitorMagic	0,367	0,171	0,196
Spiceworks	0,595	0,336	0,259
Tembria server monitor	3,233	0,736	2,497
ManageEngine	3,324	0,550	2,774
SolarWinds	3,921	1,775	2,146
ActiveXperts	4,707	1,134	3,573
PA server monitor	7,318	1,176	6,142
WebWatchBot	12,205	0,591	11,614
Ipsentry	12,776	6,992	5,784
ServerAssist	94,827	18,805	76,021

Table. 4. Bandwidth results detail

C. *DISK*

As mentioned before, this is a very important part in the benchmark. We do not want to lose any of the records written or read by the SQL Database server. We can see MonitorMagic is in the top 5 tools that use the least disk performance.

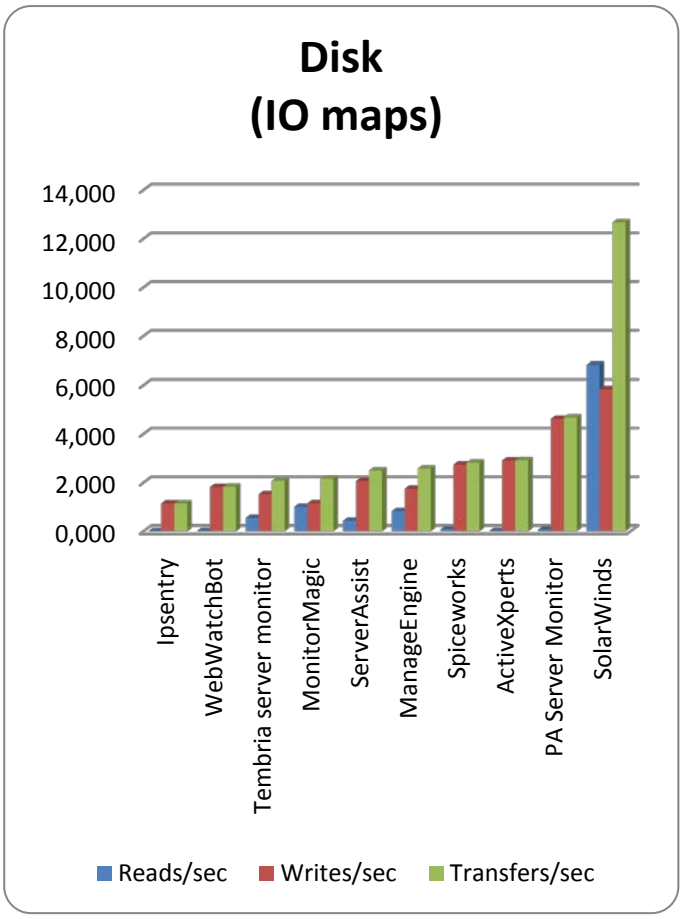


Fig. 3. Disk results

With the details listed in the following table

monitor	Reads /sec	Writes /sec	Transfers /sec
Ipsentry	0,000	1,146	1,146
WebWatchBot	0,013	1,816	1,829
Tembria server monitor	0,549	1,522	2,071
MonitorMagic	1,009	1,150	2,159
ServerAssist	0,430	2,068	2,498
ManageEngine	0,826	1,752	2,578
Spiceworks	0,079	2,738	2,817
ActiveXperts	0,008	2,910	2,917
PA Server Monitor	0,062	4,620	4,682
SolarWinds	6,832	5,839	12,671

Table. 5. Disk results detail

D. *Price*

The price is a parameter that may not be underestimated. Good tools come with high prices, especially when it comes to implementing the tool.

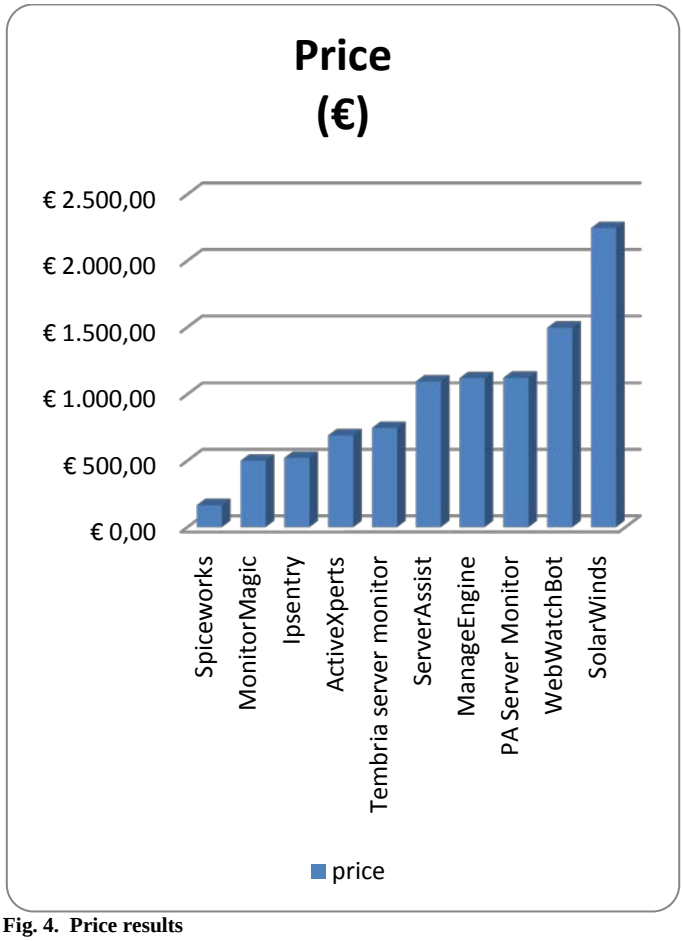


Fig. 4. Price results

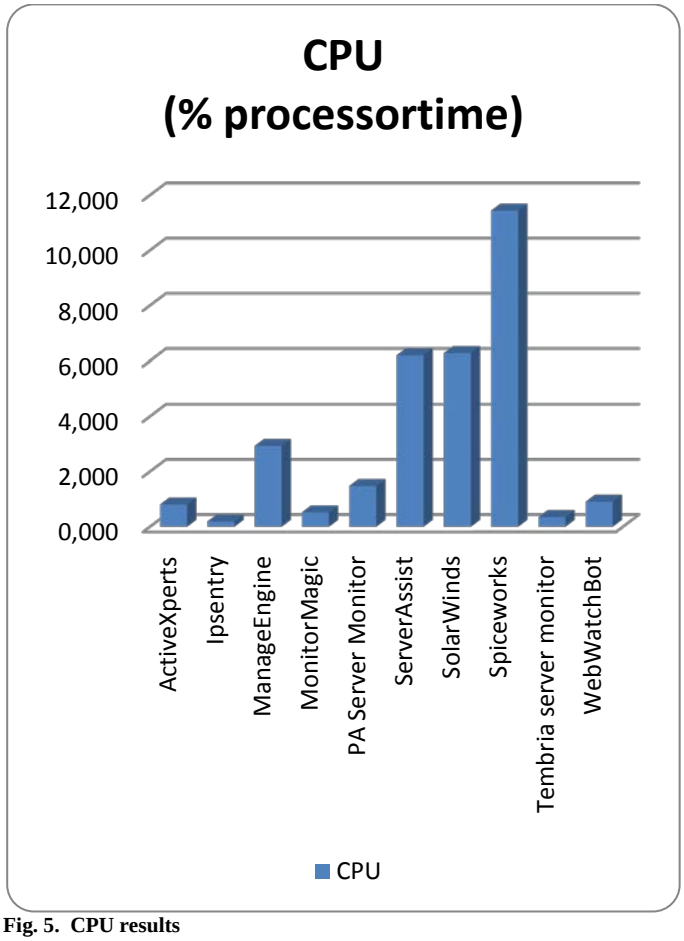


Fig. 5. CPU results

With the details listed in the following table

monitor	Price
Spiceworks	€ 164,92
MonitorMagic	€ 499,00
Ipsentry	€ 520,99
ActiveXperts	€ 690,00
Tembria server monitor	€ 745,88
ServerAssist	€ 1.095,00
ManageEngine	€ 1.120,69
PA Server Monitor	€ 1.123,69
WebWatchBot	€ 1.495,50
SolarWinds	€ 2.245,13

Table. 6. Price results detail

E. CPU

This parameter is less important, because of the high performance of modern servers this will not be a problem.

With the details listed in the following table

monitor	CPU
Ipsentry	0,189
Tembria server monitor	0,351
MonitorMagic	0,522
ActiveXperts	0,806
WebWatchBot	0,908
PA Server Monitor	1,475
ManageEngine	2,930
ServerAssist	6,193
SolarWinds	6,276
Spiceworks	11,441

Table. 7. CPU results detail

F. Memory

This sections covers the same result as CPU, modern servers have enough memory so it wouldn't cause any problem.

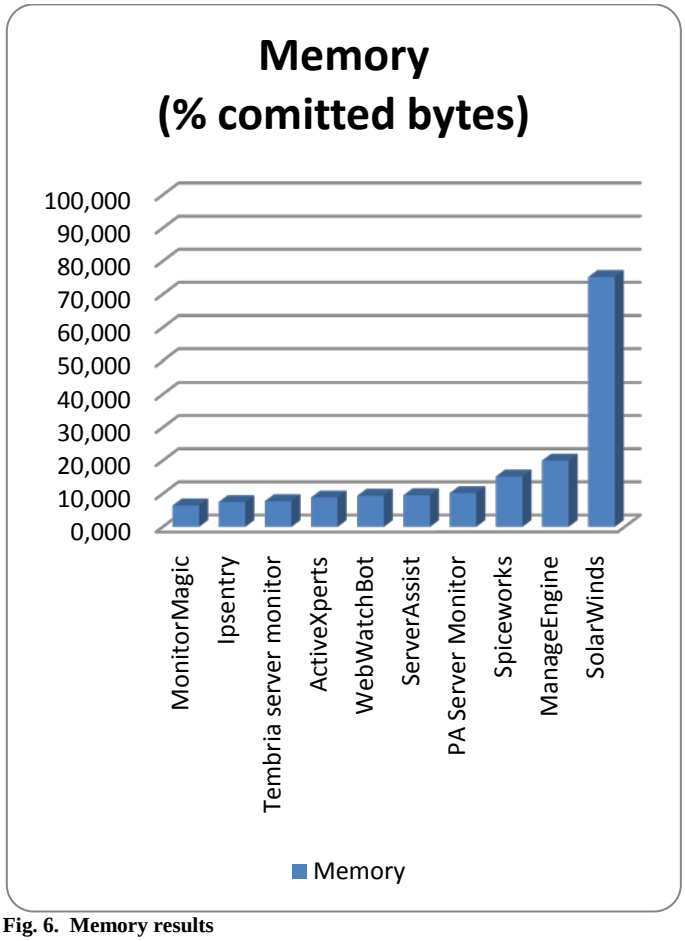


Fig. 6. Memory results

With the details listed in the following table

monitor	Memory
MonitorMagic	6,436
Ipsentry	7,492
Tembria server monitor	7,742
ActiveXperts	8,865
WebWatchBot	9,380
ServerAssist	9,526
PA Server Monitor	10,170
Spiceworks	15,171
ManageEngine	19,970
SolarWinds	75,218

Table. 8. Memory results detail

IV. Conclusion

After excessive testing in a standardized environment, we have come up with the best tool that competes with the requirements. Conclusions can be taken in several departments:

- Network load
- Disk
- Price
- CPU
- Memory

The summarization consists of mean values of all measured results, classified by importance in decreasing order and listed from best to worst. All of this gives us the best suitable tool for the company.

As you can see in the benchmark section, there is a great difference concerning network load, DISK, CPU, memory and the price that comes with the tool.

The most important factors were discussed earlier, that brings us to the overall comparison of the tools and their performance.

The following graph arranged according to best performance to worst will give us the best suitable tool for the company. A small remark concerning the graph, the price will not be listed in the graph because of the scale. When we embed the price in the overall comparison the differences between network load, DISK, CPU and memory will not be visible. The price is already mentioned in the benchmark section.

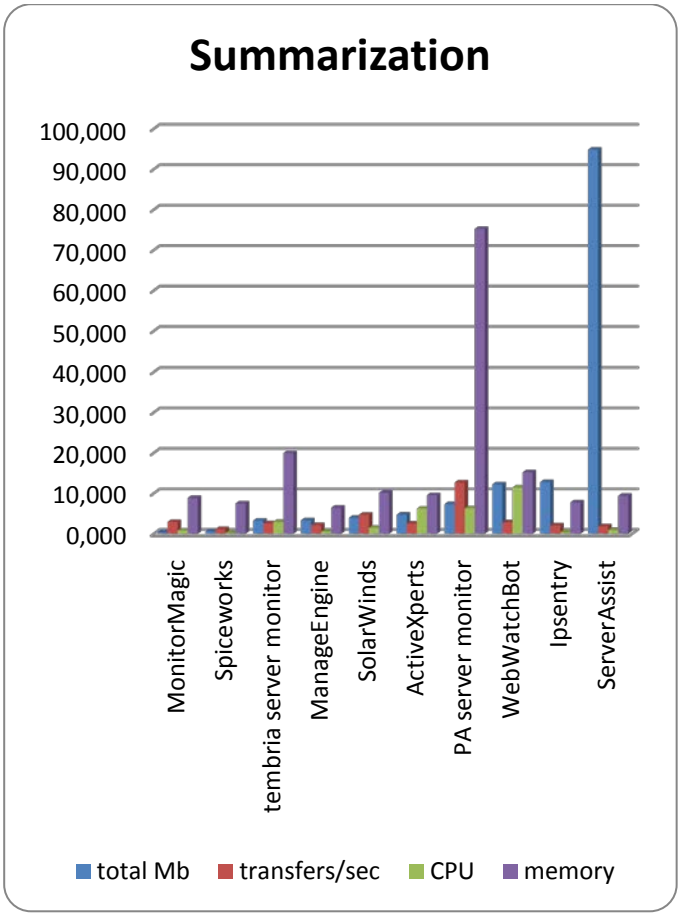


Fig. 7. Summarization results

When we bring all this together, as well as taking a look at the ease of use. MonitorMagic is the most suitable tool for Rescotec.

This brings us to testing it in a working network, which gives approximately the same results as mentioned before. We can conclude that we found the solution for the downtime of servers in the company without frequently checking the parameters.

ACKNOWLEDGMENT

First of all, I would like to thank Rescotec for giving all the necessary materials for testing and doing the research. Also a special thanks to Joan De Boeck for helping me with benchmark problems and correcting this paper.

REFERENCES

[1] Alwin Brokmann. “Monitoring Systems and Services”. Computing in High Energy and Nuclear Physics, La Jolla, California, March 2003.

[2] MICROSOFT CORPORATION. Windows 2000 professional resource kit. http://microsoft.com/windows2000/library/resources/reskit/, 2000.

[3] MICROSOFT CORPORATION. Monitoring performance. http://www.cwlp.com/samples/tour/perfmon.htm, 2001.

[4] JAY BAELE, 2007. Wireshark & Ethereal network protocol analyzer toolkit. Syngress Publishing Inc, Rockland, 523p.

The implementation of wireless voice through picocells or Wireless Access Points

Jo Van Loock [1], Stef Teuwen[2], Tom Croonenborghs[3]
3: Department of biosciences and technology Department, KH Kempen University College, Geel

***Abstract*—**

Poor coverage in buildings and ensuring a good quality became the biggest problems of voice communication and are the major cause that business customers change their provider.

To have a maximum coverage and quality for wireless voice communication one can use Picocells or Wireless Access Points (WAP's).

Picocells will enable voice communication through the normal Public Switched Telephone Network (PSTN) while WAP's will use the advancing Voice over Internet Protocol (VoIP) technology. The choice many network designers have to make is to use picocells or VoIP technology to ensure an optimal coverage and quality in voice traffic. This choice is mostly made based on a site survey. Nevertheless, the advantages and disadvantages of both solutions need to be known and considered. Sometimes network designers can consider skipping the site survey and make the choice only based on experience in the field.

I. INTRODUCTION

Ever since 1876 people have been using voice communication technology to communicate with each other. It was made possible with the efforts of Alexander Graham Bell and Thomas Watson. In 1907, Lee De Forest made a revolutionary breakthrough by inventing the three-way vacuum tube. This allowed an amplification of signals, both telegraphic and voice. By the end of 1991 the generation of mobile phones was introduced to the world. This made mobile communication, over the still developing telephone network also known as Public Switched Telephone Network (PSTN), possible. The next couple of years the problem of poor coverage and ensuring good quality of voice communications kept growing and are nowadays the major causes of business customer churn (*churn*: the process of losing customers to other companies since switching providers is done with the utmost ease).

Network designers need to be able to make a choice to resolve this specific problem. The two major solutions are the use of picocells or WAP's with implementing the VoIP protocol.

Firstly most network designers make a site survey. This step will ensure that the designer comprehends the specific radio frequency (RF) behavior, discovers RF coverage areas and checks for objects that will have a certain RF interference. Based on this data, he can make appropriate choices for the placements of the devices. Also very important is to know the advantages and disadvantages of both options so that in some cases the cost of making a site survey can be eliminated for the designing process.

Let us explain this using a small example:

If a network designer needs to implement a wireless network in a certain building and he knows the different advantages and disadvantages of both implementations, he can choose between the placement options solely on experience. This will result in a lower cost of implementation. Suppose, he would choose for the WAP implementation, knowing that a WAP costs 200 to 300 € and a complete site survey of the complex would cost 5000 to 7000 €. In this case, it would be cheaper to just add a few WAP's here and there to ensure maximum coverage over a certain area then doing the survey. The downside here is that the designer will never know the RF behavior in the complex what can lead to rather clumsy situations when a problem arises. Some problems are not knowing where coverage holes are or areas of excessive packet loss. The same example can be made with the use of picocells.

II. RESEARCHING POSSIBLE IMPLEMENTATION OPTIONS

A. Picocells

To extend coverage to indoor areas where outdoor signals do not have a good reach, it is possible to use picocells to improve the quality of voice communication. These cells are designed to provide the coverage in a small area or to enhance the network capacity in areas that have a dense phone usage. A picocell can be compared to the cellular telephone network. It converts an analogous signal to a wireless one.

The key benefits of picocells are:

- They generate more voice and data usage and supports major customers of the operator with the best quality of service.
- They reduce *churn* and drive traffic from fixed lines to mobile networks.
- They make sales of new services possible; even with improving macro cell performance.
- They prevent more costs to the infrastructure through 'Pinpoint Provisioning'; adding coverage and capacity precisely where it's needed.
- They provide a flexible, low impact and high performance solution that integrates easily with all core networks.

B. VoIP through WAP's

VoIP services convert your voice into a digital signal that travels over an IP-based network. If you are calling a traditional phone number, the signal is converted to a traditional telephone signal before it reaches its destination. VoIP allows you to make a call directly from a computer, a VoIP phone, or a traditional analog phone connected to a special adapter. In addition, wireless "hot spots" that allow you to connect to the Internet, might enable you to use VoIP services.

The advantages that drive the implementation of VoIP networks are[1][2]:

- **Cost savings**: Using the PSTN network will result in bandwidth that is not being used, since PSTN uses TDM that dictates a 64 kbps bandwidth per voice channel. VoIP shares bandwidth across multiple logical connections. Hereby we get a more efficient use of the available bandwidth. Combining the 64 kbps channels into high-speed links we need a vast amount of equipment. Using packet telephony we can multiplex voice traffic alongside data traffic which results in savings on equipment and operations costs.
- **Flexibility**: An IP network will allow more flexibility in the pallet of products that an organization can offer their customers. Customers can be segmented which helps to provide different applications and rates depending on traffic volume needs.
- **Advanced features**
    - **Advanced call routing**: e.g.: Least-cost routing and time-of-day routing can be used to select the optimal route for each call.
    - **Unified messaging**: This enables the user to do different tasks all in one single user interface. e.g.: read e-mail, listen to voice mail, view fax messages, …
    - **Long-distance toll bypass**: Using a VoIP network, we can circumvent the higher fees that need to be paid when making a trans-border call.
    - **Security**: Administrators can ensure that IP conversations are secure in an IP network. Encryption of sensitive signaling header fields and massage bodies protect packets in case of unauthorized packet interception.
    - **Customer relationships**: A helpdesk can provide customer support through the use of different mediums such as telephone, chat, e-mail. Hereby the customer satisfaction will increase.

In the traditional PSTN telephony network, it is clear to an end user which elements are required to complete a call. When we want to do a migration to VoIP, we need to be aware and have a thorough understanding of certain required elements and protocols in an IP network.

VoIP includes these functions:

- **Signaling:** To establish, monitor and release connections between two endpoints, generating and exchanging control information is necessary. This is done by signaling it. To do voice signaling, we need the capability to provide supervisory, address and alerting functionality between nodes. VoIP presents several options for signaling like H.323, Media Gateway Control Protocol (MGCP), Session Initiation Protocol (SIP)[3]. We can do signaling through a peer-to-peer signaling protocol, like H.323 and SIP, or through a client/server protocol, like MGCP. Peer-to-peer signaling protocols Peer-to-peer signaling protocols have endpoints that have onboard intelligence that enables them to interpret call control messages, and initiate and terminate calls. Client/server protocols on the other hand lack the control intelligence but communicate to a server (call-agent), by sending and receiving event notifications
  For example: When a MGCP gateway determines a telephone that has gone off hook, it does not know to give a dial tone automatically. In this case the call agent informs the gateway to provide a dial tone, after the gateway has send an event notification to it.
- **Database service**: includes access to billing information, caller name delivery, toll-free database services and calling card services.
- **Bearer control**: Bearer channels are channels that carry voice calls. These channels need a decent supervision so that appropriate call connect and call disconnect signaling be passed between end devices.
- **Codecs**: the job of a codec is the coding and decoding translation between analog and digital devices. The voice coding and compression mechanism used for converting voice streams, differs for every codec.

C. Implementation type choice

With careful consideration to both implementation methods which enable mobile communication we opted in favor of placing multiple WAP's and enabling VoIP protocol on the network. The implementation cost of using WAP will be considerably higher in comparison with picocells but the expenses of making telephone calls internally, will considerably decrease.

Above a decrease of the call cost, the improved security, explained in the advanced features section above, was also a decisive factor for making this choice.

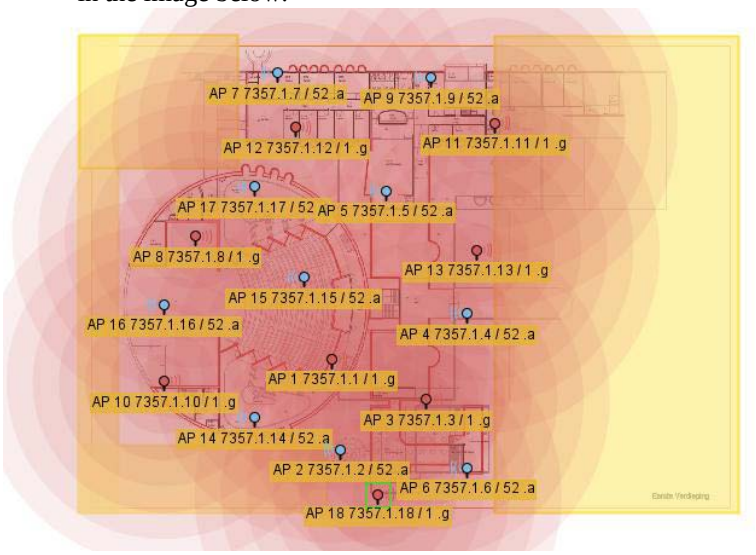
D. Site survey

The choice about the type of implementation was made purely on experience at "De Warande". Therefore I opted to make a small site survey on my own. Hereby I used the following steps[4] to perform my site survey:

1. Obtain a facility diagram in order to identify the potential RF obstacles.
2. Visually inspect the facility to look for potential barriers or the propagation of RF signals and identify metal racks.
3. Identify user areas that are highly used and the ones that are not used.
4. Determine preliminary access point (AP) locations. These locations include the power and wired network access, cell coverage and overlap, channel selection, and mounting locations and antenna.

5. Perform the actual surveying in order to verify the AP location. Make sure to use the same AP model for the survey that is used in production. While the survey is performed, relocate AP's as needed and re-test.
6. Document the findings. Record the locations and log of signal readings as well as data rate at outer boundaries.

Using the steps mentioned above I firstly made a theoretical site survey (step 1-4), through the use of Aruba RF plan, of every floor - 5 floors in building A, 6 floors in building B. This program is able to pin point the optimal WAP locations on a certain floor, where we need the 802.11 a/b/g wireless coverage, without including the interference of concrete walls or thick glass and irradiation from other levels. This is shown in the image below:

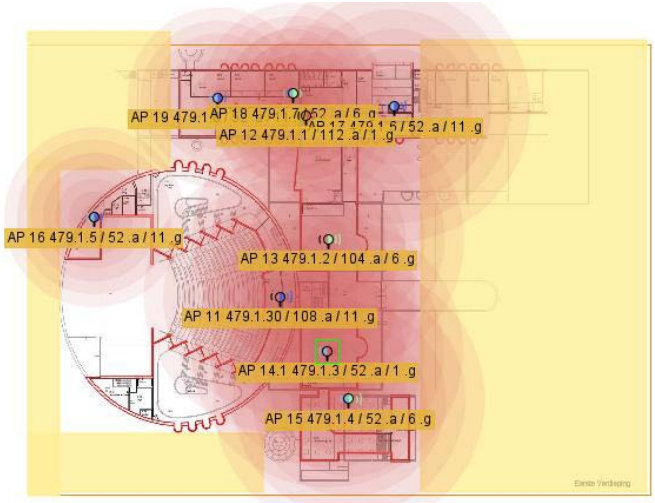


After this theoretical approach of the floor we need to do actual surveying on site to verify the WAP locations and make proper adjustments when needed. During the survey we need to allocate possible problems. When located, we consider the possible level of interference it will cause and adjust the locations of the WAP's. Another adjustment we need to consider is the irradiation from levels below when we are dealing with open areas, since the closed areas won't have any irradiation through the thick concrete walls of the building.

When we send data, through connecting to the WAP's, we will use the 2.4-GHz or 5-GHz frequency ranges. The 2.4-GHz range is used by 802.11b and 802.11g IEEE standards and is probably the most widely used frequency range. In this range we have 11 channels, each 22MHz wide. This means that we can only use channel 1, 6 and 11 because the other channels will overlap with others and cause interference. This is one more factor we need to include when we make our actual survey. The 5.0-GHz frequency range contains the IEEE standard 802.11a. Because 802.11a uses this range and not the 2.4-GHz range it is incompatible with 802.11 b or g. 802.11a is mostly found in business networks due to the higher cost. Each standard has its pros and cons[5]:

- 802.11a pros:
  - Fast maximum speed (up to 54 Mbps)
  - Regulated frequencies prevent signal interference from other devices
- 802.11a cons:
  - Highest cost
  - Shorter signal range that is easily obstructed
- 802.11b pros:
  - Lowest cost
  - Good range that is not easily obstructed
- 802.11b cons:
  - Slowest maximum speed (up to 11 Mbps)
  - Possibility of interference of home appliances
- 802.11g pros:
  - Fast maximum speed (11Mbps using DSSS and up to 54 Mbps using OFDM)
  - Good signal range that is not easily obstructed
  - Uses OFDM to gain bigger data rates
  - Backward compatible with 802.11b
- 802.11g cons:
  - More expensive then 802.11b
  - Possibility of interference of home appliances

At the Warande we opted to use all three standards. This way we are sure that, there will always be enough open connections for clients. This is of no inconvenience to the client since the present technology of wireless network adapters will search for a connection regardless of the standard being used (when supported). The result is shown in the image below:



The yellow areas in the image represent areas where there is no need for coverage or areas where we do not care if there is coverage or not.

Using this method I was able to conclude that there are 16 WAP's needed in the first building to provide the areas with enough coverage for wireless internet connection and 3 extra WAP's to ensure the needed coverage for voice traffic. The second building needed 13 WAP's to get enough coverage for the wireless internet connections and an additional 14 WAP's for the necessary coverage for voice traffic.

III. The Configuration

Since the need for security in the sector is very high, I will explain this section by means of a few examples, because I can not share the actual configuration method and commands

with the public.

The configuration needed, must allow a person to call internally to other IP phones or to analog phones externally. Also we must foresee usage of faxes. This means that a configuration of analog ports for the faxes and digital ports for the actual calls is necessary. Next to these two different methods, we also have to consider some factors that influence making a design.

A. Factors that influence Design

When we use VoIP, we are sending voice packets via IP. Hereby it is normal that certain transmission problems will popup. Because the listener needs to recognize and sense the mood of the speaker, we need to be able to minimize the effect of these problems. The following factors[1] can affect clarity:

- **Echo**: result of electrical impedance mismatches in the transmission path. Effecting components are the amplitude (loudness) and delay (time between spoken voice and the echo). Echo is controlled by using suppressors or cancellers.
- **Jitter**: variation in the arrival of coded speech packets at the far end of a VoIP network. This can cause gaps in the playback and recreation of the voice signal.
- **Delay**: time between the spoken voice and the arrival of the electronically delivered voice at the far end. Delay results from distance, coding, compression, serialization and buffers.
- **Packet Loss**: Under various conditions like unstable network, congestion, voice packets can be dropped. This means that gaps in the conversation can get perceptible to the user.
- **Background noise**: low-volume audio that is heard from the far-end connection.
- **Side tone**: the purposeful design of the telephone that allows the speaker to hear their spoken audio in the earpiece. If side tone is not available, it will give the impression that the telephone is not working properly.

Some simple solutions for these problems are:

- Using a priority system for voice packets.
- Using dejitter buffers.
- Use codecs to minimize small amounts of packet loss
- Making a minimized congestion network design

Since we need to minimize these specific factors we will use Quality of Service (QoS). QoS is deployed at different points in the network. With implementing this we will have a certain voice section that is protected from data-bursts.

Two other subjects that influence design are knowing the amount of bandwidth needed for voice traffic and how we can reduce overall bandwidth consumption.

Because WAN bandwidth is the most expensive bandwidth there is, it would be useful to compress the data we have to send. This will be done by a specific codec, for example: G.711, G.728, G.729, G.723, iLBC, … .

The codec that is used at the Warande is the G.729 codec. This codec uses Conjugate Structure Algebraic Code Excited Liner Prediction (CS-ACELP) compression to code voice into 8kbps streams. G.729 has two annexes A and B. G.729a requires less computation, but lowering the complexity of the codec is not without a trade-off because the speech quality is marginally worsened. Also G.729b adds support for Voice Activation Detection (VAD) and Comfort Noise Generation (CNG), to cause G.729 to be more efficient in its bandwidth usage. If we take a bundle of approximately 25 calls or more, 35% of the time will be silence. In a VoIP network whether it is a conversation or silence, it is packetized. VAD can suppress packets containing silence. With interleaving data traffic with VoIP conversations the VoIP gateways will use network bandwidth more efficiently. A silence in a call can be mistaken for being disconnected. This is also solved with VAD since it provides CNG. CNG will make the call appear normally connected to both parties by generating white noise locally.

Voice sample size is a variable that can affect the total bandwidth used. To reduce the total bandwidth needed, we must encapsulate more samples per Protocol Data Unit (PDU = is the control information that is added at each layer of the OSI-model, when encapsulation occurs.) But larger PDU's will risk causing variable delay and several gaps in communication. That is why we use the following formula to determine the number of encapsulated bytes in a PDU, based on the codec bandwidth and the sample size.[2]

*Bytes_per_sample = (Sample_Size * codec_Bandwidth) /8*

Meaning, if we would use the G.729 codec, and knowing that the standard for sample size is 20 bytes -and the bandwidth for G.729 is 8kHz this would result in:
Bytes_per_sample = (0.020 * 8000) /8 = 20

Another characteristic that influences the bandwidth is the layer 2 protocol used to transport VoIP. Depending on the choice of the protocol, it is possible that the overhead will grow substantially When the overhead is higher, the bandwidth needed for VoIP will increase as well. Depending on what security measures or the kind of tunneling used, the overhead will also increase.
For example: Using a virtual private network, IP security will add 50 to 57 bytes of overhead. Considering the small size of a voice-packet this amount of overhead is a significant amount. All these factors, codec choice, data-link overhead, sample size, … have positive and negative impacts on the total bandwidth. To calculate the total bandwidth that is needed we must consider these contributing factors as part of the equation[2]:

- More bandwidth required for the codec requires more total bandwidth.
- More overhead associated with the data link requires more total bandwidth.
- Larger sample size requires less total bandwidth.
- RTP header compression requires significantly less total bandwidth. (*RTP defines a standardized packet format for delivering audio and video over the internet. It includes a data portion and a header portion. The header portion is much larger than the data portion since it contains an IP segment, UDP segment and a RTP segment. Standard = 40 bytes of overhead uncompressed and 2 to 4 bytes compressed*)

Considering these factors the calculation to calculate the total bandwidth required per call is done with the following formula [2]

*Total_Bandwidth = ([Layer2_overhead + IP_UDP_RTP_overhead + Sample_Size] / Sample_Size) * Codec_Speed*

Meaning if we use a G.729 codec, 40-byte sample size, using Frame Relay with Compressed RTP it would result in:
Total_Bandwidth = ([6 + 2 + 40] / 40) * 8.000 = 9.600 bps

If we would have no RTP compression it becomes:
Total_Bandwidth = ([6 + 40 + 40] / 40) * 8.000 = 17.200 bps
When we take the utilization of VAD into account on both examples:
Total_Bandwidth = 9.600 – 35% = 6.240 bps
Total_Bandwidth = 17.200 – 35% = 11.180 bps

This shows us the great advantage of using the G.729 codec that supports VAD.

B. Configuring Analog Ports

For a long time analog ports were used for many different voice applications such as: local calls, PBX-to-PBX calls, on-net / off-calls, etc. Now that we only work with digital phones we only connect our fax machines to the analog ports.

Faxes are something completely different as to making a simple telephone call. Fax transmissions operate across a 64 kbps pulse code modulation (PCM) encoded voice circuit. In packet networks on the other hand, the 64 kbps stream is in most cases compressed to a much smaller data rate. This is done by using a codec that is designed to compress and decompress human speech. Fax tones deviate from this procedure and therefore a sort of relay or pass-through mechanism is needed. There are three available options to operate fax machines in a VoIP network[2]:

1. **Fax relay:** The fax bits are demodulated at the local gateway, the information is send across the voice network using the fax relay protocol and finally the bits are remodulated back into tones at the far gateway. The fax machines are unaware that a demodulation/modulation fax relay is occurring. Mostly the packetizing and encapsulating of data is done by the ITU-T T.38 standard and is available for H.323, MGCP and SIP gateway control protocol.
2. **Fax pass-through:** The modulated fax information from the PSTN is passed in-band with an end-to-end connection over a voice speech path in an IP network. There are two pass-through techniques:
    a. The configured codec is used for voice and fax transmission. This is only possible using the G.711 codec with no VAD en Echo cancellation (EC) or when a clear channel codec is used like G.726/32. In this case the gateways make no difference between voice and fax calls. Two fax machines communicate with each other completely in-band over a voice call.
    b. Codec up speed or fax pass-through with up speed method. This means that the codec configured for voice is dynamically changed to the G.711 codec by the gateway. The gateways are to some extent aware that a fax call is made by recognizing a fax tone, automatically changing, through the use of Named Signaling Event (NSE) messaging, the voice codec to G.711 and turn off EC and VAD for the duration of the call.

    Fax pass-through is supported by H.323, MGCP and SIP gateway control protocol.
3. **Fax store-and-forward:** This method breaks up the fax process in sending and receiving processes. For incoming faxes from the PSTN, the router will act as an on-ramp gateway. Here the fax will be converted to a Tagged Image File Format (TIFF) file which will be attached to an e-mail and forwarded to the end-user. For outgoing faxes the router will act as an off-ramp gateway, where an e-mail with a TIFF attachment will be converted to a traditional fax format and delivered to a standard fax machine. The converting is done with the ITU-T T.37 standard.

The choice that was made for the Warande was to use Fax pass-through with up speed. This choice was made because the equipment was not suited for the fax store-and-forward option. On the other side the fax relay method was not chosen because the available bandwidth was not an issue. The choice of using up speed was because almost the whole network uses codec G.729, which is incompatible for using the first pass-through method.

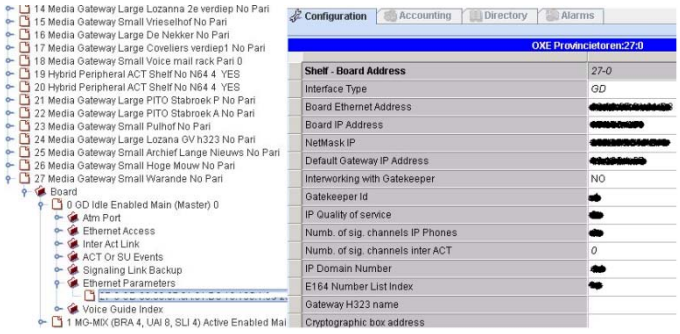
C. Configuring Digital Ports

Digital circuits are used when interconnecting the VoIP network to the PSTN or to a Private Branch Exchange (PBX). The advantage of using digital circuits is the economies of scale made possible by transporting multiple conversations over a single circuit.

Since the “Provincie Antwerpen” has a contract with Belgacom as their telecom operator, they use the Integrated Services Digital Network (ISDN) network for their calling services. The equipment used supports the ISDN Basic Rate Interface (BRI) and ISDN Primary Rate Interface (PRI). Both media types uses B and D channels, where B channels carry user data and D channels will direct the switch to send incoming calls to particular timeslots on the router[6]. Normally the PRI will be used to make PBX-to-PBX calls or other internal calls and the BRI will be used when a connection to an outside network is made.

At the Warande, it is a little different. There are 8 BRI interfaces to connect to the outside world. Since every BRI supports 2 channels, the Warande can make 16 outgoing calls at the same time. When for example a 17th user wants to make an outside call, he will be routed around the network to Antwerp. Here he will be connected to the telephone central that will give him an outside connection on their BRI interface. Now that the outside calls can be made we have to make sure we can do internal calls. This is done using a call system that is purely based on IP. All the calls will travel over the network as voice packets that will be protected by configuring a Quality of Service (QoS).

Configuring the BRI and internal IP network is not done the way students learn it. Because we are configuring and managing a large amount of sites and an even larger amount of phone devices it would be too much trouble doing the installation with a console program. Instead, we use OmniVista 4760. This allows us to have an efficient control over all sites and on the other hand we can make changes with a few clicks. A screenshot of the program can be found below. Here we can see a couple of sites that are managed by the program.


14 Media Gateway Large Lozanna 2e verdiep No Pari
15 Media Gateway Small Vrieselhof No Pari
16 Media Gateway Large De Nekker No Pari
17 Media Gateway Large Coveliers verdiep1 No Pari
18 Media Gateway Small Voice mail rack Pari 0
19 Hybrid Peripheral ACT Shelf No N64 4 YES
20 Hybrid Peripheral ACT Shelf No N64 4 YES
21 Media Gateway Large PITO Stabroek P No Pari
22 Media Gateway Large PITO Stabroek A No Pari
23 Media Gateway Small Pulhof No Pari
24 Media Gateway Large Lozana GV h323 No Pari
25 Media Gateway Small Archief Lange Nieuws No Pari
26 Media Gateway Small Hoge Mouw No Pari
27 Media Gateway Small Warande No Pari
Board
0 GD Idle Enabled Main (Master) 0
Atm Port
Ethernet Access
Inter Act Link
ACT Or SU Events
Signaling Link Backup
Ethernet Parameters
Voice Guide Index
1 MG-MIX (BRA 4, UAI 8, SLI 4) Active Enabled Mai
Configuration
Accounting
Directory
Alarms
OXE Provincietoren:27:0
Shelf - Board Address
27-0
Interface Type
GD
Board Ethernet Address
Board IP Address
NetMask IP
Default Gateway IP Address
Interworking with Gatekeeper
NO
Gatekeeper Id
IP Quality of service
Numb. of sig. channels IP Phones
Numb. of sig. channels inter ACT
0
IP Domain Number
E164 Number List Index
Gateway H323 name
Cryptographic box address

D. VoIP gateways and gateway control protocols[3]

To provide voice communication over an IP network, dynamic Real-time Transport Protocol (RTP) sessions are created and formed by one of many call control procedures. Typically, these procedures integrate mechanisms for signaling events during voice calls and for handling and reporting statistics about voice calls. There are three protocols that can be used to implement gateways and make call control support available for VoIP:

1. H.323
2. Media Gateway Control Protocol (MGSP)
3. Session Initiation Protocol (SIP)

As mentioned earlier, the "Provincie Antwerpen" uses a peer-to-peer signaling strategy. This means that MGCP, which is client/server signaling can be removed from the available protocols. That leaves us with H.323 and SIP. H.323 is the gateway protocol used at the Warande or any other Provincial site. The reason is subject to the different implementations of equipment.

For example: The main site in Antwerp has three different kinds of telephone centrals: a state of the art one and two older ones. All these centrals need to be able to communicate with each other and if we would use SIP on one of them the others need to be able to support the same protocol. Which in this case is impossible. All the centrals do support H.323, which gives us the reason why this protocol has been used.

IV. Conclusion

The problem was to solve the poor coverage at "De Warande" and ensuring a good quality of voice communication. This is possible by the use of picocells that enable voice communications through the normal PSTN network or by using WAPs with the VoIP protocol.

The choice made for "De Warande" is to use a certain number of WAPs placed at strategic places. These spots where calculated through experience and making a small site survey to measure and comprehend the RF behavior of the site.

With the choice made the next thing on the "to do" list was to configure the network. Here we needed to watch out for some factors that have a negative influence on the design such as echo, jitter, delay, … . Also a measurement of the total bandwidth, that was needed for our voice traffic to travel on, was calculated. When the preparations were made there were two different things we had to do.

Firstly there was the configuration of analog ports. These ports were used to connect fax machines into the network. We discussed the three possibilities that could be used for enabling the faxing mechanism. The fax pass-through method was the one selected.

Secondly, the configuration of digital ports was completed. These port interfaces are mostly used for making connections to the PSTN network or to a PBX. The configuration of the digital ports was done using an ISDN PRI and ISDN BRI interface. The PRI was used for internal purposes and BRI for connecting to the outside world.

Finally we searched for a suitable gateway protocol. These protocols will dynamically create and facilitate RTP sessions to provide voice communication of an IP network. Here were three major protocols available, H.323, MGCP and SIP. We easily excluded MGCP from the list, being a client/server protocol. Afterwards SIP was also excluded through the different implementations of equipment.

References

[1] Staf Vermeulen, *Course IP-telephony Master ICT*.
[2] Kevin Wallace, *Authorised Self-Study Guide Cisco Voice over IP (CVOICE) Third Edition*, Cisco Press, First Print 2008, 125-183 + 185-244
[3] Denise Donohue, David Mallory, Ken Salhoff, *Cisco IP communications Voice Gateways and gatekeepers*, Cisco Press, Second printing 2007, 25-52+53-78 + 79-114
[4] http://www.cisco.com
[5] Staf Vermeulen, *Course CCNA 4: Accessing the WAN, Master ICT*
[6] Patrick Colleman, *Course Datacommunicatie, Master ICT*

Usage sensitivity of the SaaS-application of IOS International

Luc Van Roey[1], Piet Boes[2], Joan De Boeck[1]
[1]IIBT, K.H. Kempen (Associatie KILeuven), B-2440 Geel, Belgium
[2]IOS International, Wetenschapspark 5, B-3590 Diepenbeek, Belgium
luc_van_roey@hotmail.com, piet.boes@iosint.be, joan.deboeck@khk.be

***Abstract*— Software as a service (SaaS) is one of the latest hypes in the mainstream world. The quality of a SaaS-application is assessed in terms of response time. An inferior quality of a SaaS-application can lead to frustrated users and will eventually create lost business opportunities. On the other hand, company expenditures for a SaaS-infrastructure are linked with the application's expected traffic. In an ideal infrastructure, we want to spend just enough, and not more, allocating resources to get the most beneficial result. This paper tries to identify the reaction of the SaaS-application of IOS International with different user loads and to assess if the SaaS-application meets the expectations of it's clients. Eventually we'll see that the response time is directly proportional to the user loads as long as there are no errors in the user loads. We show also that the actual infrastructure meets the expected response time for an application load of 10 editors and 90 viewers.**

***Index Terms*— SaaS, load testing, IOS Mapper, response time**

I. Introduction

IOS International nv, a Belgium Company, develops a software platform IOS to increase the productivity and the quality of risk management within an organization.
A new objective of IOS International is to make their software available on the Internet as Software as a Service (SaaS). This way the customer no longer has to buy the software, but only concludes a contract for the services that he needs.

Software as a Service (SaaS) is one of the latest hypes in the mainstream world. The quality of a SaaS-application is assessed in terms of response time. An inferior quality of a SaaS-application can lead to frustrated users and will eventually create lost business opportunities. On the other hand, company expenditures on a SaaS-infrastructure are linked with the application's expected traffic. In an ideal infrastructure we want to spend just enough, and not more, allocating resources to get the most beneficial result. [1]

Load testing offers the possibility of measuring the performance of the SaaS-application based on real user behavior. This behavior is imitated by building an interaction script with the user requests. A load generator, like Jmeter, then passes through the interaction script, adapted with test parameters based on a real-life environment, on the SaaS-application IOS Mapper.

With these load tests we can identify the reaction of the SaaS-application IOS Mapper with different user loads and assess if the SaaS-application IOS Mapper meets the expected real-life user loads.

II. Response time

As mentioned in the introduction, the quality of the SaaS-application IOS Mapper can be measured in terms of response time. So it will be very important to monitor these end-to-end response times to stipulate how long it lasts before the requests of the user are carried out and will be visible for the user. Afterwards we can compare these results with frustration level times.

From studies (Nah, 2004) into acceptable answer times it becomes clear: [2]

- Delay of 41 seconds is suggested as the cut-off for long delays like downloading reports;
- Delay of 30 seconds is suggested as the frustration level for long delays;
- Delay of 12 seconds causes satisfaction to decrease for normal actions like opening wizards.

III. Load testing

Load testing offers the possibility of measuring the performance of the SaaS-application based on the real user behavior. This behavior is imitated by building an interaction script with the user requests. A load generator, like Jmeter, then passes through the interaction script, adapted with test parameters based on a real-life environment, on the SaaS-application IOS Mapper.

The load generator imitates the behavior of the web browser: it sends continuous requests to the SaaS-application,

waits a certain time after the SaaS-application has answered to the request (this is the thinking time which real users also have) and then sends a new request. The load generator can simulate thousands of concurrent users at the same time to test the SaaS-application.

We'll use Jmeter as the load generator. It's a completely free Java desktop application. With Jmeter we'll record the behavior of the users of the SaaS-application IOS Mapper. Afterwards we'll make a load model from the recordings. We can introduce this load model into Jmeter and subsequently we are able to simulate our load model.

Each simulated web browser is a virtual user. A load test will only be valid if the behavior of the virtual users resembles the behavior of the effective users. For this reason the behavior of the virtual users must

- follow patterns resembling real users;
- use realistic thinking times
- have an asynchronous behavior between each user.

Figure 1 shows a load model of a virtual user using the SaaS-application IOS Mapper, based on the patterns of a real-life user. [3]

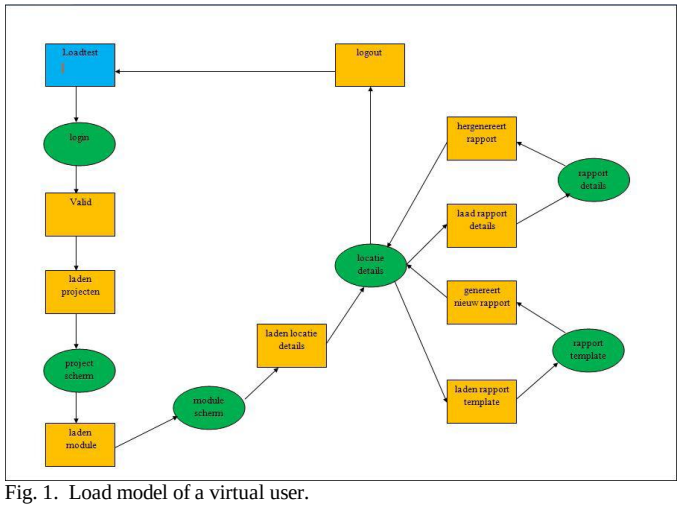


Fig. 1. Load model of a virtual user.

Each rectangle in figure 1 represents the requests that a user sends to the SaaS-application IOS Mapper. The SaaS-application will respond to these requests and this will eventually lead to a visible window in the user's web browser. This corresponds with the green ellipse in figure 1.

IV. USAGE SENSITIVITY OF IOS MAPPER

A. *Single user*

In the first test it's the intention to find the minimum response times of the SaaS-application. One virtual user will pass through the complete load model which can be seen in figure 1.

If the user wants to generate a report, it takes a response time of 13 seconds. This is the longest transaction, as shown in figure 2. The generation of a report will be the most important reason for delays and crashes.

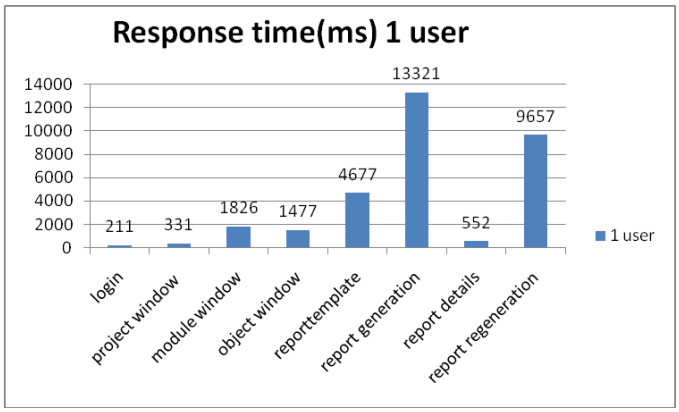


Fig. 2. Response time of 1 user

Furthermore it's also important to know if the end-to-end response times are influenced by the pc or the bandwidth of the Internet of the users. For this test we used a AMD Athlon 64 X2 Dual Core Processor 4200+GHZ with 2.00GB RAM as pc and a AMD Turion 64 Mobile 1.99GHZ with 1.00GB RAM as laptop. The laptop is significantly slower than the pc. We will also use the laptop on 2 different locations with its own Internet. The first location has Internet with a bandwidth of 4Mbit and the second location has Internet with a bandwidth of 12Mbit.

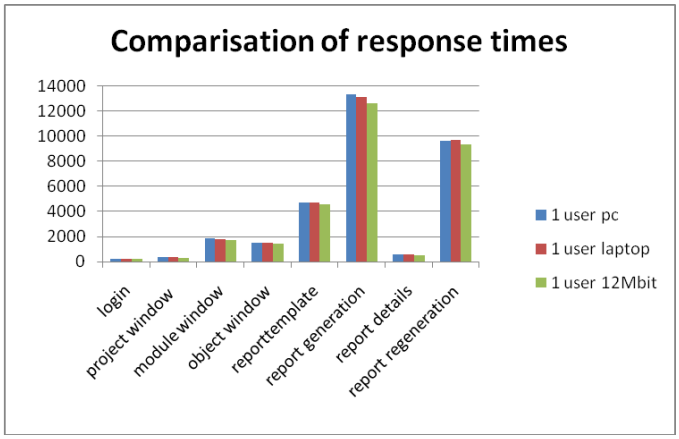


Fig. 3. Response time of 1 user

Figure 3 shows that there is no difference between the usages of a slower or a faster pc. If we raise the bandwidth of the Internet from 4Mbit to 12Mbit, we measure a small difference of 5percent. This difference isn't significant enough and is unimportant.

B. *Several simultaneous users*

In figure 2 we see that the response times for report generation are the longest. In the following test we measure the response time for the generation of reports when more

and more simultaneous users pass through the load model seen in figure 1.

Up to 25 simultaneous users, there's an increase in duration of the response time directly proportional to the increase of the user loads when generating a report. This is shown in figure 4. We also notice that out of 25 users there are some users who will get an error in answer to their request, because the server can't process the load.

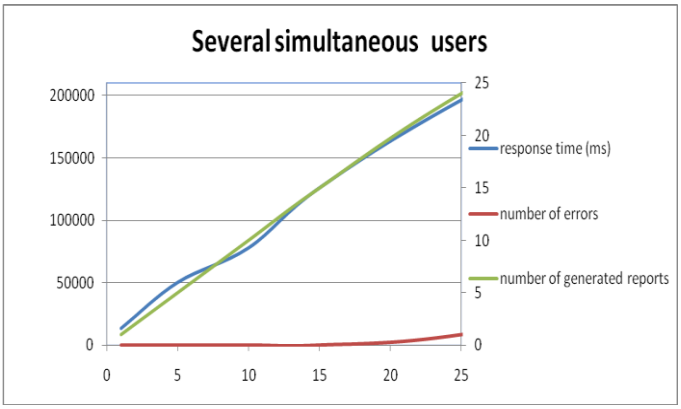


Fig. 4. Several simultaneous users up to 25 users

If we raise the number of simultaneous users up to 100 users, we can see in figure 5 that there will be a logarithmic increase in the duration of the response times. The reason for this is that the number of users that receive an error on their request grows exponentially. This means that with 100 users there are not 100 users who generate a report, but only 65 of them. If we take this into account and in the graph we only show the users who effectively generate a report, again we will get a directly proportional increase, as shown in figure 6.

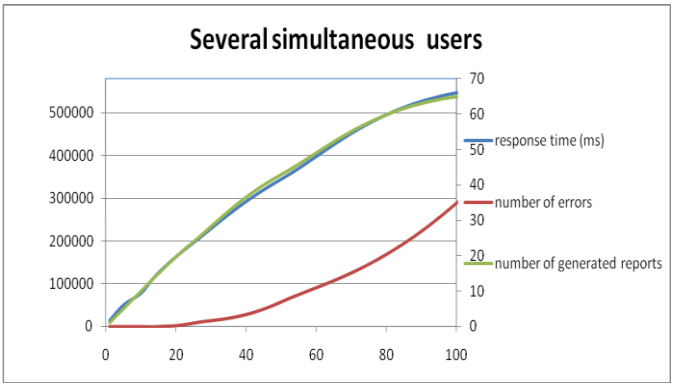


Fig. 5. Several simultaneous users up to 100 users

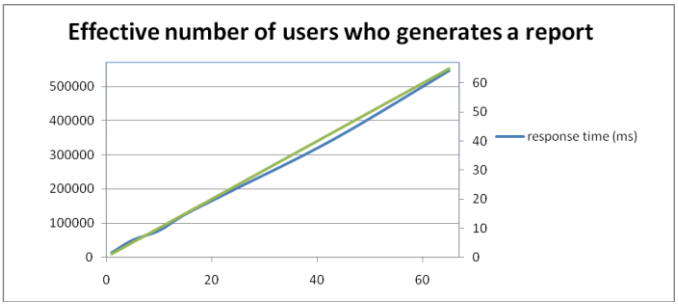


Fig. 6. Effective number of users who generate a report

If we further increase the number of users, we can see in figure 7 that the SaaS-application IOS Mapper can generate a maximum of 67 reports simultaneously. This will result in an average response time of 580 seconds or 9.5 minutes. We can also see that the number of users that effectively generates a report without receiving an error, decreases from this point on to 700 users. From this point on no-one can generate a report anymore. The server won't respond to anything.

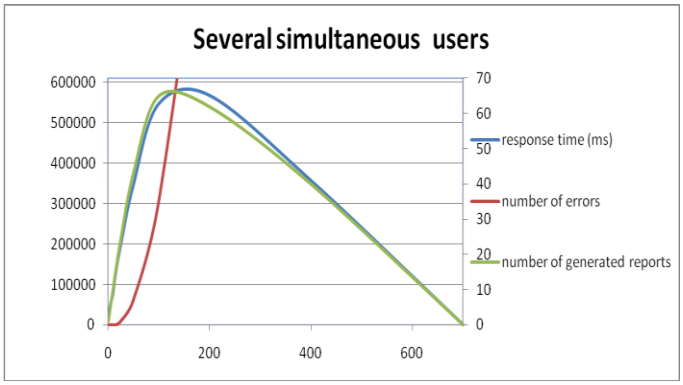


Fig. 7. Several simultaneous users until the server crashes

In figure 8. we tested if there was a difference in response time between the load test on 1 pc and on 2 pc's with each its own Internet.

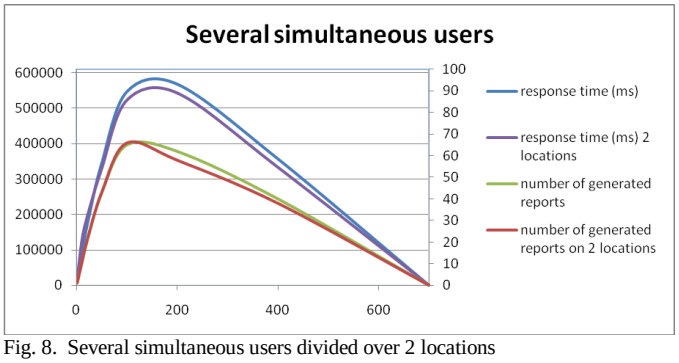


Fig. 8. Several simultaneous users divided over 2 locations

It's clear that the bandwidth has no influence on the end-to-end response times of the SaaS-application IOS Mapper.

C. Real-life approach

In reality, several users will never carry out the same request simultaneously and the consecution of requests will never immediately follow each other without time delay. Each user will use the SaaS-application at a different moment. And each user has a thinking time for completing an action. These thinking times will be different for each action and will differ for every user. These things have to be taken into account to create a real-life multi-users profile.

If we only take the abovementioned values into account, then the SaaS-infrastructure will be too powerful for the number of users that can use the SaaS-application. This

ensures that the bulk of the investments in the SaaS-infrastructure isn't totally exploited.

In these circumstances a maximum of 25 users can use the SaaS-application without a user experiencing errors. Optimum use of the SaaS-application IOS Mapper would allow even less than 5 users. The generation of a report takes an average of 50 seconds and the opening of a wizard lasts 11 seconds, as shown in figure 9. As explained above in II. RESPONSE TIME, a user will shut down the SaaS-application IOS Mapper and won't make use of it anymore, leading to commercial loss.

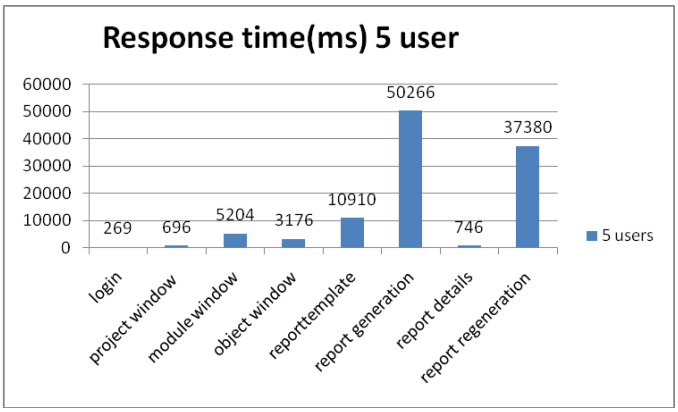


Fig. 9. Response time 5 simultaneous users

A real-life multi-users profile of IOS International is shown in figure 10. At the moment there are 10 editor users and 90 viewers.

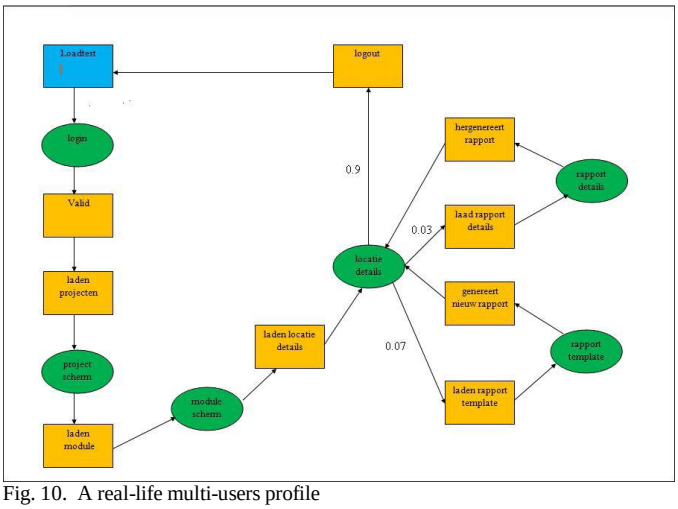


Fig. 10. A real-life multi-users profile

After bringing the thinking times into account, we get the following response times, as shown in figure 11.

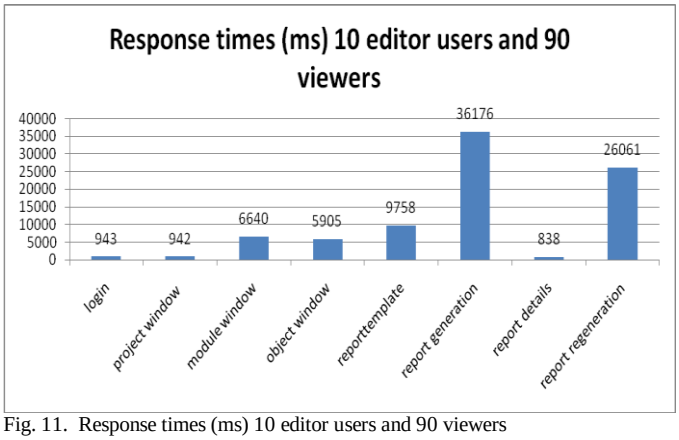


Fig. 11. Response times (ms) 10 editor users and 90 viewers

We see that the response time of the report template takes around 10 seconds and the generation of a report around 36 seconds. This falls within the frustration standards explained above in II. RESPONSE TIME.

V. Conclusion

The SaaS-application IOS Mapper of IOS International is independent of the quality of a contemporary pc used on the client side and the SaaS-application is also independent of the bandwidth of the used Internet, in assumption that every user has a broadband Internet.

As the IOS Mapper application will be more heavily loaded, the response time will increase directly proportional to the user loads. We showed also that the actual infrastructure meets the expected response time for an application load of 10 editors and 90 viewers. This is the current clientele, but it will quickly expand in the future.

Due to the directly proportional increase between the response time and the user loads, IOS International can, at conscription of new clientele, stipulate the expected response time and intervene prematurely to improve the SaaS-infrastructure without overpowering the infrastructure.

These load tests can also be used in the future to control new updates of IOS Mapper. A sudden increase of response time for a certain request under the same user load indicates a bug in the application. These bugs can then be fixed in advance, without the user having to face these bugs.

Acknowledgment

I want to thank Brigitte Quanten for the linguistic advice.

References

[1]Yunming, P., Mingna, X. (2009). Load Testing for web applications. *First International Conference on Information Science and Engineering*, 2954-2957.

[2]Nah, F. (2004). A study on tolerable waiting time: how long are Web users willing to wait? *Behaviour and Information Technology,* 23(3), 153-163

[3]Grundy, J. Hosking, J. Li, L., Liu, N. *Performance Engineering of Service Compositions.* PowerPoint presentation, The University of Auckland. Founded at: http://conferenze.dei.polimi.it/SOSE06/presentations/Hosking.pdf

Fixed-Size Least Squares Support Vector Machines Study and validation of a C++ implementation

S. Vandeputte, P. Karsmakers

***Abstract*— We propose an implementation in C++ of the Fixed-Size Least Squares Support Vector Machines (FS-LSSVM) for Large Data Sets algorithm originally developed in MATLAB. An algorithm in MATLAB is known to be suboptimal with respect to memory management and computational performance. These limitations are the main motivation for a new implementation in another programming language.**

First , the theory of Support Vector Machines is shortly reviewed in order to explain the Fixed-Size Least Squares variant. Next the mathematical core of the algorithm, which is solving a linear system, is zoomed into. As a consequence we explore a set of LAPACK implementations for solving a set of linear equations and compare in terms of memory usage and computational complexity. Based on these results the Intel MKL library is selected to be included in our new implementation. Finally, a comparison in terms of computational complexity and memory usage is performed on a MATLAB and C++ implementation of the FS-LSSVM algorithm.

***Index Terms*—Fixed-Size Least Squares Support Vector Machines, kernel methods, LAPACK, C++,**

I. INTRODUCTION

In this work an optimized implementation in C++ for the large-scale machine learning algorithm called Fixed-Size Least Squares Support Vector Machines (FS-LSSVM), which was proposed in [1], is presented. Although this algorithm was already found competitive with other state-of-the-art algorithms, no detailed discussion about an optimal implementation was studied. This paper concerns the latter since an optimal program might result in handling even larger data sets on the same computer system.

The FS-LSSVM algorithms resides in the family of algorithms which all are strongly connected to the popular Support Vector Machines (SVM) [2] which is the current state-of-the-art in pattern recognition and function estimation. Least-Squares Support Vector Machines (LS-SVM) [3][4] simply the original SVM formulation. While SVM boils down to solving a Quadratic Programming (QP) problem, the LS-SVM solution is found by solving a linear system.

Using a current standard computer[1] the LS-SVM formulation can be solved for large-data set problems up to 10.000 data points using of the Hestenes-Stiefel conjugate gradient algorithm [5][6]. In order to solve an even larger set of problems with sizes up to 1 million of data points an approximate algorithm called FS-LSSVM was proposed in [4]. In [1] this algorithm was further refined and compared to the state-of-the-art. The authors there programmed the algorithm in MATLAB. Such an implementation is known to be suboptimal with respect to memory usage and computational performance. This is due to the fact that MATLAB is a prototyping language which enables fast algorithmic development but has the limitation that the resources cannot be accessed with full control.

In this work we aim at a new FS-LSSVM implementation which provides solutions for the above limitations.

The paper is organized as follows. In Section I we explained the need for a new implementation of FS_LSSVM. But first will we in Section II give a small introduction to FS-LSSVM. In section III we will introduce LAPACK and select some candidates for a performance test. Section IV explains some technical details about the test. Section V will handle the test results. Finally in Section VI we will implement the algorithm of which we will present the performance result in Section VII.

II. FIXED-SIZE LEAST SQUARES SUPPORT VECTOR MACHINES

In this section we will give a short introduction to LSSVM regarding classification. The following steps are the same for regression.

According to Suykens en Vandewalle[3], the mentioned optimization problem for classification becomes in primal weight space

$$\min_{w,b,e} \Im(w,e) = \frac{1}{2}\langle w, w\rangle + \frac{\gamma}{2}\sum_{k=1}^{n} e_k^2$$

$$\text{with } Y_k\left[\langle w, \varphi(X_k)\rangle + b\right] = 1 - e_k \qquad k = 1,\ldots,n \quad .$$

[1] E.g. a computer with an Intel Core2Duo processor

The classifier in primal weight space takes the form

$$y(x) = sign\left(\langle w, \varphi(x) + b\rangle\right)$$
met $w \in \Re^{n_h}$
$b \in \Re$.

After using Lagrange multipliers, the classifier can be computed in dual space and is given by

$$y(x) = sign\left(\sum_{k=1}^{n} \alpha_k Y_k K(x, X_k) + b\right)$$
with $K(x, X_k) = \langle \varphi(x), \varphi(X_k)\rangle$

α and b are solutions of the linear systeem

$$\left(\begin{array}{c|c} 0 & Y^T \\ \hline Y & \Omega + \frac{1}{\gamma} I_n \end{array}\right)\left(\begin{array}{c} b \\ \hline \alpha \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 1_n \end{array}\right)$$
with $1_n = (1, \ldots, 1)^T$
$\Omega_{kl} = Y_k Y_l \langle \varphi(X_k), \varphi(X_l)\rangle$

and a positive definite kernel

$$K(X_{k,X_l}) = \langle \varphi(X_k), \varphi(X_l)\rangle \quad .$$

It would be nice if we could solve the problem in primal space but then we need an approximation of the feature map. We can handle this through random active selection with Renyi entropy criterium. After this Nyström approximation we have a sparse prediction comparison.

$$y(x) = \langle w, \tilde{\varphi}(x)\rangle + b$$
met $w \in R^m$.

With that featuremap approximation we can then solve a ridge regression problem in primal space with the a sparse representation of the model, which is the core of the FS-LSSVM algoritme.

III. LAPACK

The mathematical core of FS-LSSVM is finding the solution for a system of linear equations. A general available standard software library for solving linear systems is the Linear Algebra PACKage (LAPACK). It depends on another library the Basic Linear Algebra Subprograms (BLAS) to effectively exploit the caches on modern cache-based architectures. Many different implementations of the LAPACK and BLAS library combination are available. In order to be able to solve the linear system as fast as possible it is worth the investigation to find out the best performing implementation.

Four known LAPACK and BLAS implementations were tested:

- Mathworks MATLAB R2008b: MATLAB makes use of a LAPACK implementation, for Intel CPUs the Intel Math Kernel Library v7.0. The test may reveal the influence of MATLAB as LAPCK wrapper.
- Reference LAPACK v3.2.1: libraries which are reference implementations of the BLAS [9] and LAPACK [10] standard. These are not optimized and not multi-threaded, so a bad performance is to be expected.
- Intel Math Kernel Library (MKL): implementation of Intel which of course exploits the most out of Intel processors. Version 10.2.4 is used.
- GotoBlas2: a BLAS library completely tuned at compile time for best performance on the CPU it is compiled on.

Of course there are more LAPACK implementations available than the ones we selected for testing. For some reason they were left out like e.g; ACML is the AMD implementation while only test on Intel processors.

IV. TEST

We developed a test application to solve the equation Ax=B, in C++ using LAPACK functions `dgesv()` for double precision and `sgesv()` for single precision input data, in MATLAB using the operator "\" or mldivide function. During the lifetime of a software application dynamic memory (which is used to store the matrices A and B) can get fragmented. To make sure fragmentation is as low as possible for using the biggest possible array sizes; we locate and allocate the two biggest chunks of contiguous memory immediately at the start of the test. These two memory blocks are used to store the matrices A and B, which increase during the lifetime of the test to do a performance test of different sizes until a row size of 10000.

While it is sufficient to compare different implementations based on their time spent, it may be useful to compare the theoretical and achieved performance. The ratio between achieved performance F and theoretical peak performance P_{peak} is known as efficiency [7]. A high efficiency indicates an efficient numerical implementation. Performance is measured in floating point operations per second (FLOPS) and can be calculated as

$$P_{peak} = n_{CPU} * n_{core} * n_{FPU} * f$$

with n_{CPU} the number of CPUs in the system, n_{core} the number of computing cores per CPU, n_{FPU} the number of floating point units per core and f is the clock frequency. The achieved performance F can be computed as the flopcount divided by the time. For the xgesv() function of

LAPACK is the standard number of floating point operations 0.67 * N3 [8].

Intel CPU	P_{piek} (GFLOPS) double	P_{piek} (GFLOPS) float
Pentium D 940	12,8	25,6
Core2Duo E6300	14,88	29,76
Xeon E5506	34,08	68,16

Table 1 Intel microprocessor export compliance metrics.

The value of n_{FPU} is an estimation of the number of units. By the use of SIMD (Single Instruction Multipe Data) instruction has a processor the ability to do processing in parallel and do not have real FPU's anymore. Depending on the architecture some constant values that are more or less correct are agreed upon.
When using floating point precision (4 bytes) in stead of double precision (8 bytes) the processor can handle twice as many datainstructions because of the bytesize.

We will test the performance of the mentioned solvers on different CPU architectures of Intel as these are a good representative of the x86 family CPUs on the market today. Chosen architectures are:

- "Netburst": used in all Pentium 4 processors and a Pentium D 920 @ 3,20 GHz as test CPU.
- "Core": lower frequency but more efficient than the "Netburst", chosen CPU is a Core2Duo E6300 @ 1,86 GHz
- "Nehalem": has a focus on performance with a Xeon E5506 @ 2,13 GHz to test.

All test are performed on Windows XP SP3 operating system.

V. LAPACK RESULTS

Two kind of results are available, de time performance results and the efficiency results.

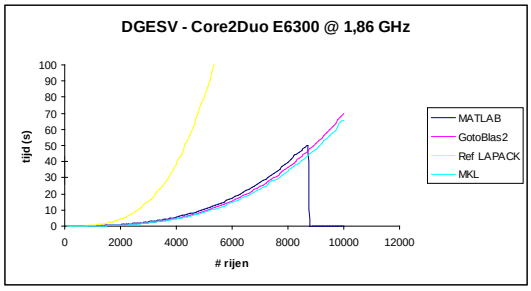


Figure 1 Time results of LAPACK

In Figure 1 is there an immediate result visible, the performance of ther reference Lapack is rather bad, actually the curve is $O(N^3)$. We can also see that Matlab cannot handle more dan 8300 sized matrices, due to lack of memory or good memory management inside the application. The libraries GotoBlas2 and MKL are close to each other.

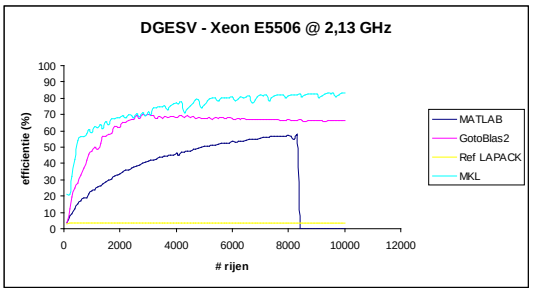


Figure 2 Efficiency results of LAPACK

Concerning the efficiency results, lets have a look at Figure 2. The conclusion of Figure 1 is definitely confirmed and now we see more clearly that the MKL library has a better performance than GotoBlas2.
There is also a remarkable conclusion about GotoBlas2 when you look at all the figures (Appendix A).
On older architectures GotoBlas2 is better than MKL, on newer architectures with more cores and larger caches GotoBlas2 is less performant but also it is degrading when the matrix size rises.

For the C++ implementation of FS-LSSVM, we will use MKL as LAPACK library.

VI. IMPLEMENTATION

We will handle the implementation in C++ in this section of the paper.

There are 4 important requirements we must try to realize during this new development:

- Memory usage: we have to keep the overhead over redundant data as low as possible. Goal is having an algorithm that can handle larger matrices than with MATLAB. We will deal with this requirement by using pointers of C++.
- Performance: We hope we dealt with it by choosing the most performant LAPACK library.
- Datatype: it would be nice if the algorithm would also work for floats in stead of doubles. Then one can test the accuracy of floats compared to doubles, if floats would be accurate enough than FS-LSSLVM can handle larger matrices. This requirement will be fulfilled when we use C++ templates.
- Code maintenance: It is very import to keep de code structure as equal as possible with the MATALB code. Changes in the original algorithm can than easily be transferred to the new code.

VII. IMPLEMENTATIONRESULTS

We are going to compare the different implementations with regards to time.

We picked randomly some datasets from [11] and used them as inputdata for the two algoritms. Test were performed on the Pentium D 940.

testnaam	#inputdata	MATLAB (s)	FSLSSVM++(s)	%
testdata	120	1,85	0,55	0,30
mpg	392	7,57	1,83	0,24
australian	690	20,27	5,97	0,29
abalone	4177	202,60	45,56	0,22
mushrooms	8124	1575,14	344,88	0,22

Figure 3 MATLAB – FS_LSSVM t.o.v. FSLSSVM++

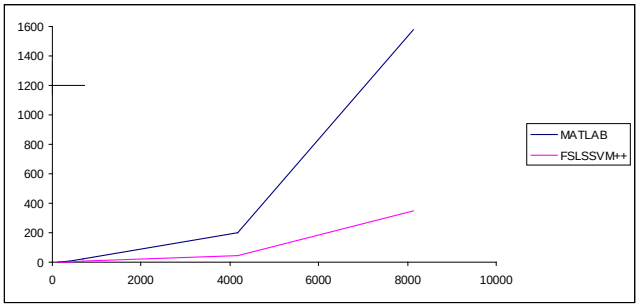


Figure 4 MATLAB – FS_LSSVM t.o.v. FSLSSVM++

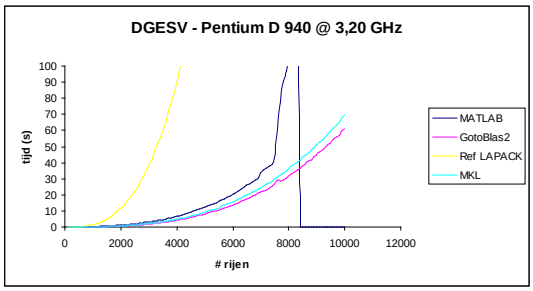
Even we did only some random tests and the algorithm can react differently according to the inputdata, the results are much better than expected. We can state that the new implementation is 70 % better dan the MATLAB code.

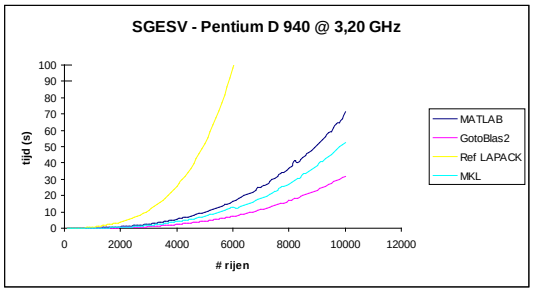
REFERENCES

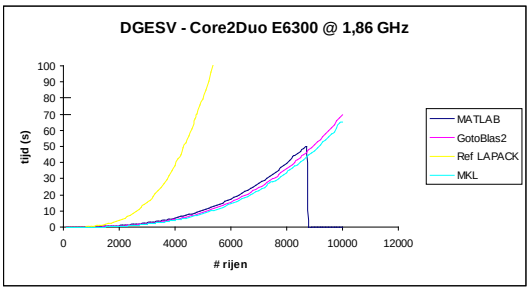
[1] K. De Brabanter, J. De Brabanter, J.A.K. Suykens, B. De Moor, Optimized Fixed-Size Least Squares Support Vector Machines for Large Data Sets, 2009.
[2] V. Vapnik, Statistical Learning Theory, 1999
[3] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, 1999
[4] J.A.K. Suyskens et al, Least squares support vector machines, 2002
[5] G. Golub, C. Van Loan, Matrix computations, 1989
[6] J.A.K. Suykens et al , Least squares support vector machine classifiers : a large scale algorithm,1999
[7] T. Wittwer, Choosing the optimal BLAS and LAPACK library., 2008
[8] LAPACK benchmark, “Standard” floating point operation counts for LAPACK drivers for n-by-n matrices, http://www.netlib.org/lapack/lug/node71.html#standardflopcount
[9] C.L. Lawson, et al, Basic Linear Algebra Subprograms for FORTRAN usage, 1979
[10] E. Anderson, et al, LAPACK users’ guide, 1999
[11] LibSVM Data: Classification, Regressin and Multi-label: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

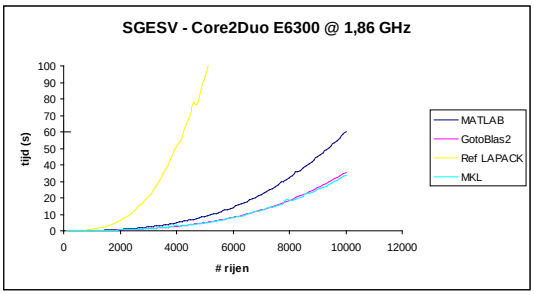
Appendix A: LAPACK results

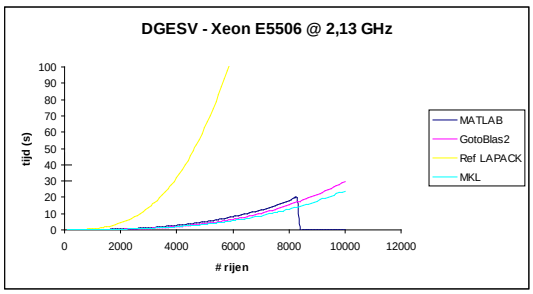
Time

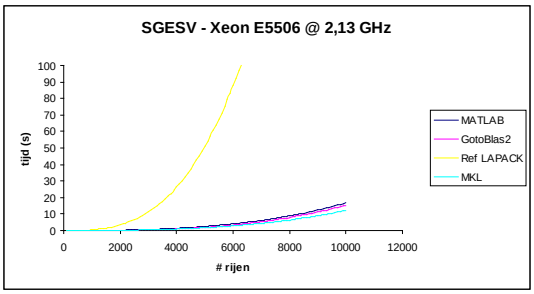




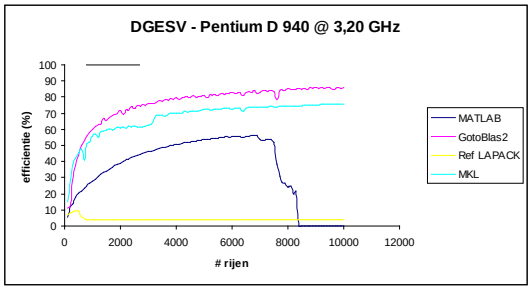


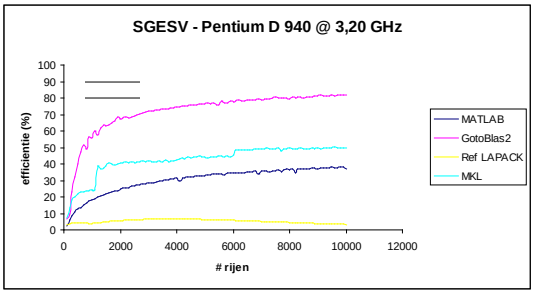


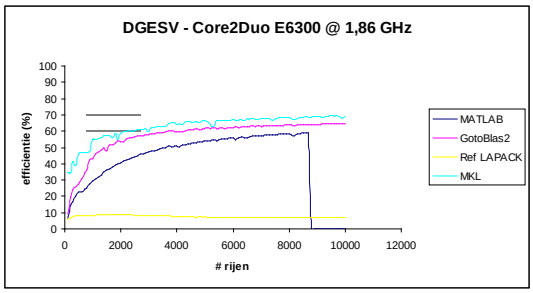


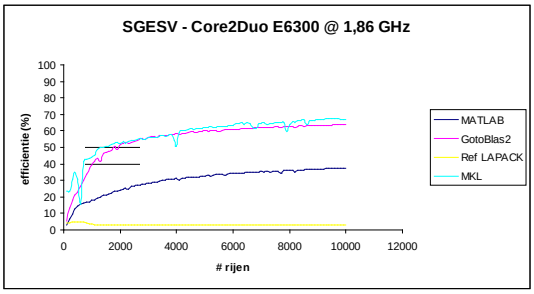


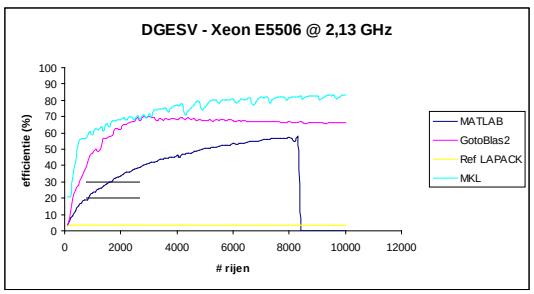
Efficiency

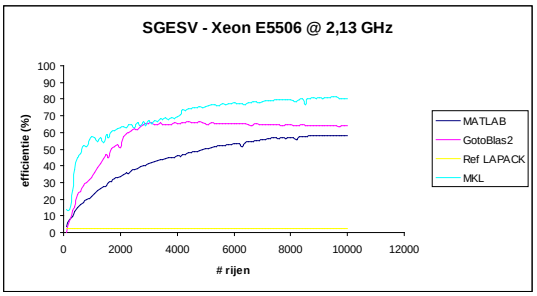











SGESV - Xeon E5506 @ 2,13 GHz
efficientie (%)
100
90
80
70
60
50
40
30
20
10
0
0
2000
4000
6000
8000
10000
12000
rijen
MATLAB
GotoBlas2
Ref LAPACK
MKL

Improving audio quality for hearing aids

P. Verlinden, Katholieke Hogeschool Kempen, peter.verlind@gmail.com
S. Daenen, NXP Semiconductors, steven.daenen@nxp.com
P. Leroux, Katholieke Hogeschool Kempen, paul.leroux@khk.be

***Abstract*—Since hearing problems are becoming more frequent these days, the necessity for high quality hearing aids will grow. In order to achieve high audio quality it is necessary to use a good audio codec. Nowadays there are a lot of high quality audio codecs, but because the target application is a hearing aid, some limitations need to be taken into consideration such as delay and hardware limitations. This is the reason why a low complexity codec like the Philips Subband Coder is used. In this paper an implementation of the Philips Subband Coder (SBC) is discussed and a comparison with the G.722 speech codec will be made.**

I. Introduction

Hearing aids have improved greatly over time. Today a lot of hearing aids are binaural. This means that the audio received on the right hearing aid will also be transmitted to the hearing aid in the left ear and vice versa. This greatly improves the hearing quality. The reason for this is simply the human brain. The brain needs both ears to determine where the sound is coming from, the distance and most importantly it helps to sort out speech from noise. In [1] benefits of binaural hearing are discussed.

In this paper a hearing aid that uses the G.722 speech codec to compress audio is discussed, this is a problem because this greatly diminishes audio quality for music signals. Therefore a better codec that also can handle music signals is searched in this paper.

Hearing aids are real-time devices and the sound received on one side must be heard on the other side with minimum delay. For this reason delay becomes a big issue. When the delay becomes too large, the person wearing the hearing aid would hear an echo, if there is no compensation by introducing buffering. Ideally it would be best to have zero delay, but since there always will be some processing delay this isn't possible. It is necessary to keep the delay as low as possible for the audio codec. A second limitation in the choice of an audio codec is the hardware. Hearing aids need to be as small as possible for high comfort. This means there isn't much space for hardware such as memory. A third limitation is battery life. A hearing aid needs a battery to operate and it isn't comfortable if the battery needs to be changed to frequently. These two limitations also imply that a low complexity codec is needed. These limitations are the reason why the Philips Subband Coder was used in this paper.

In this paper a closer look is taken at what the causes for delay are in an audio codec, since this is a very important factor for a hearing aid application. The delays from other codecs [3] than the Philips Subband Coder will be looked at. Next a closer look is taken on how the Philips Subband Coder and the G.722 codec work and a comparison is made. Next the integration of the Philips Subband Coder is discussed. After the implementation an evaluation of the audio quality is made. On the basis of this evaluation the configuration parameters are determined that are best used for the Philips Subband Coder. The results are compared with the evaluation of the G.722 codec from [4].

II. Delay introduced by codecs

Here some important elements that cause delay are discussed. Only elements that are relevant to the codecs used in this paper are discussed.

A. Filter Bank

The delay in audio codecs has many different sources. One big source of delay is the filter bank. Almost every audio codec uses a filter bank. This filter bank can be a MDCT (modified discrete cosine transform) or a QMF (quadrature mirror filter) filter. Both the Philips Subband Coder and the G.722 codec use a QMF filter bank. The delay introduced by these filter banks results from the shape and length of the filters. When calculating this delay for the Philips Subband Codec, at 32kHz sampling rate and a filter length of 80, the delay becomes 2,5 ms. It becomes clear that half of the total delay comes from the filter bank. Since the total delay for this codec is 5ms [2]. Calculating the system delay for orthogonal filter banks is done with the following formula's [3]. In these formula's N is the delay in number of samples.

$$N = filter_{length} - 1$$

$$delay = \frac{N}{f_s}$$

B. Prediction

There are two ways to use prediction in coding. Block wise prediction and backwards prediction. When using block wise prediction a block of data is analyzed. Hence the minimum delay introduced by this operation is equal to the block length. When backward prediction is used the prediction coefficients are calculated on the base of past samples. Therefore there is no delay because there's no need to wait on samples. Only the G.722 codec uses prediction, the Philips Subband Coder doesn't. But since the Philips Subband Coder also encodes the samples in blocks, a delay is introduced equal to the block length.

C. Delay in other codecs

There are a lot of codecs available these days. Since high quality for music needs to be achieved, the delay of speech codecs isn't discussed, since they perform poorly for music signals. In table 1 several codecs are listed with their

delays [3]. Notice that the lowest delay is still 20ms at a sampling rate of 48 kHz, for a sampling rate of 32 kHz this becomes even higher. For use in hearing aids this is unacceptable. The reason for this high delay is that these codecs use a psycho-acoustic model which introduces higher complexity and therefore more delay. This higher complexity means that these codecs use bigger block sizes for the encoding process, which introduces more delay. These codecs also use an MDCT filter bank. This type of filter bank also has a longer delay than a QMF filter bank.

TABLE I
OVERALL DELAYS OF VARIOUS AUDIO CODECS, SAMPLING RATE 48 KHZ

	Algorithmic delay without bit reservoir
MPEG-1 Layer-2 192 kpbs	34 ms
MPEG-1 Layer-3 128 kpbs	54 ms
MPEG-4 AAC 96 kpbs	55 ms
MPEG-4 HE AAC 56 kpbs	129 ms
MPEG-4 AAC LD 128 kpbs	20 ms

III. PHILIPS SUBBAND CODER

A. *Subband splitting*

In the first step the audio signal has to be split in several subbands. The Philips Subband Coder uses 4 or 8 subbands. To split the signal into subbands an analysis filter is used, at the decoder side a synthesis filter is used to recombine the subbands. Cosine modulated filter banks are used. Both are polyphase filter banks. These type of filters have low complexity and low delay [6,7]. For the analysis filter the modulation function is given by [2]:

$$c_k(n) = \cos\left(\frac{\pi}{M}\left(n - \frac{M}{2}\right)\left(k + \frac{1}{2}\right)\right), k \in [0,7], \quad n \in [0, L-1]$$

In this function M is the number of subbands and L represents the filter length. The synthesis filter has a similar function:

$$c_k(n) = \cos\left(\frac{\pi}{M}\left(n + \frac{M}{2}\right)\left(k + \frac{1}{2}\right)\right), k \in [0,7], \quad n \in [0, L-1]$$

B. *APCM (adaptive pulse code modulation)*

After the audio signal is split in several subbands, the samples are encoded using APCM. The first step in this encoding process is calculating scale factors. To this end, the subbands are divided in block of length 4, 8, 12 or 16. For example 128 input samples are transformed in 8*16 subband samples, which are then processed as a block. The first step is to determine the maximum value for each subband in the block. The maximum values are quantized on a logarithmic scale with 16 levels. Thus the scale factor needs 4 bits to be coded as a scale factor index. The scale factor index can be found by:

$$SFI = |\log_2(\max)|$$

After the scale factors are calculated all the samples of that block are divided by the scale factor. Such that all samples are in the interval [-1,1].

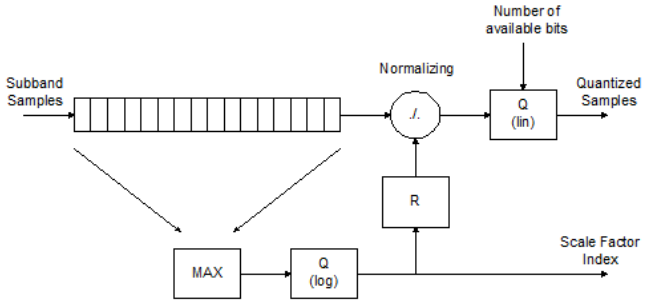


FIGURE I: APCM ENCODER

Then adaptive bit allocation is used to distribute the available bits over the different subbands. The number of bits is proportional to the scaling factor, that was calculated in the previous step. The bit allocation is based on the fact that the quantization noise in a subband can be kept equal over a 6dB range. An increase of 1 of the SFI for one band increases the quantization noise with 6dB, if one bit is added to the representation of a sample the quantization noise drops by 6dB. Thus the quantization noise can be kept constant, over all subbands, within 6dB. The bits are then distributed using a 'water-filling' method.

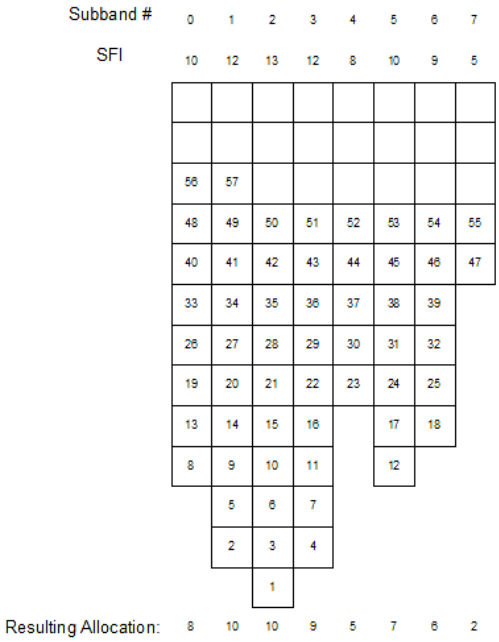


FIGURE II: WATER-FILLING

After the adaptive bit allocation the samples in each subband are quantized using the available bits assigned to each subband.

For decoding the samples, the quantized samples are multiplied with the scale factor. After this decoding these samples are sent to the synthesis filter bank.

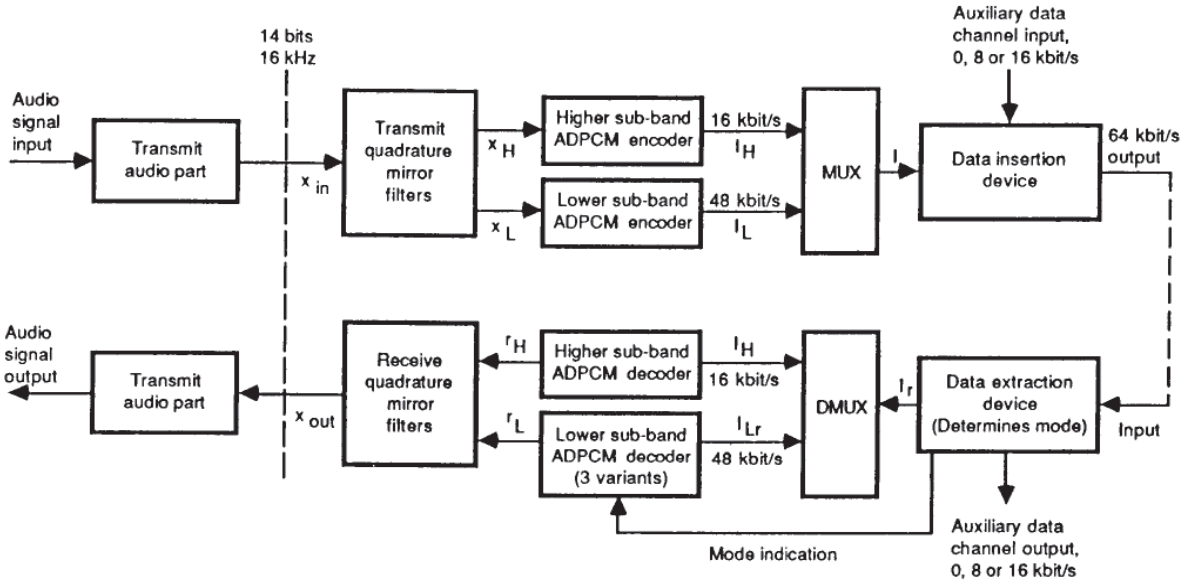


FIGURE III: G.722 BLOCK DIAGRAM

IV. G.722 CODEC

The G.722 codec as specified in [8] is used with a sampling frequency of 16 kHz. In the hardware that is used to test the Philips Subband Coder, the G.722 codec is implemented in hardware. In this setup a sampling frequency of 20.48 kHz is used. The operation of the codec is identical, the difference is that the bitrate goes up from 64 kpbs to 81.92 kpbs. Because in most cases the standard 64kpbs is used, the codec is discussed here at a sampling rate of 16 kHz.

The G.722 codec can operate in 3 modes. In mode 1 all the bits available are used for audio coding, in the other two modes an auxiliary data channel is used. Since this data channel isn't useful for this application, only mode 1 is discussed.

Figure 3 shows the block diagram for the encoder and the decoder.

A. *Quadrature mirror filters (QMFs)*

In this codec two identical quadrature mirror filters are used. At the encoder side this filter is used to split the 16 kHz sampled signal with a frequency band from 0 to 8kHz, into two subbands. These two subbands are called the lower subband (0 to 4 kHz) and the higher subband (4 to 8 kHz), these subbands are sampled at 8 kHz. These subbands are represented by the signals x_L and x_H.

The receiving QMF at the decoder is a linear-phase non-recursive digital filter. Here the signals coming from the ADPCM (adaptive differential pulse code modulation) decoders (r_L and r_H) are interpolated. The signals go from 8 kHz to 16 kHz and are then combined to produce the output signal (x_{out}) which is sampled at 16 kHz.

B. *ADPCM encoders and decoders*

In G.722 two ADPCM coders are used, one for the lower and one for the higher subband. This discussion is limited to the encoders, since this is the most important step in the coding process. For a complete overview of the decoders the reader is referred to [8].

1) *Lower subband encoder*

The lower subband encoder will produce an output signal of 48 kpbs so most of the available bits go to the lower subband. This is because G.722 is a speech codec, and most information of human speech is situated in the 0 to 4 kHz frequency band. The adaptive 60 level adaptive quantizer produces this signal. The input for this quantizer is the lower subband input signal substracted with an estimated signal. The quantizer uses 6 bits to code the difference signal.

The feedback loop is used to produce the estimate signal. An adaptive predictor is used to produce this signal. A more detailed discussion about this decoder may be found in [8]. Figure 4 show the complete block diagram of the lower subband decoder.

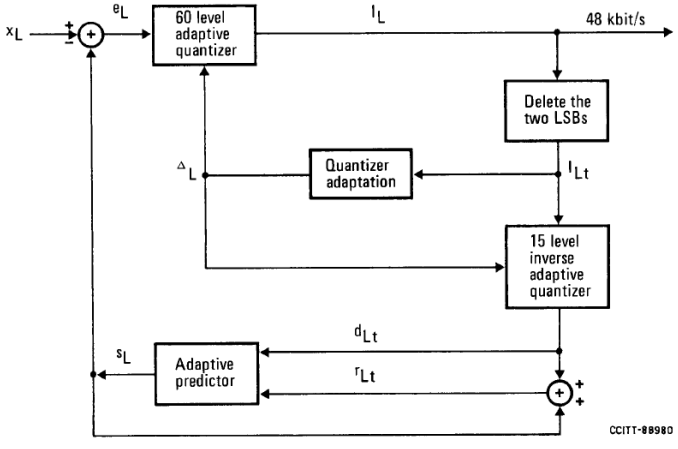


FIGURE IV: LOWER SUBBAND ENCODER

2) Higher subband encoder

The higher subband encoder produces a 16 kpbs signal. It works similarly to the lower subband encoder. The difference is that a 4 level adaptive quantizer is used instead of a 60 level quantizer. Only two bits are assigned to the difference signal. As can be seen in figure 5, the block diagram is almost identical to the lower subband decoder.

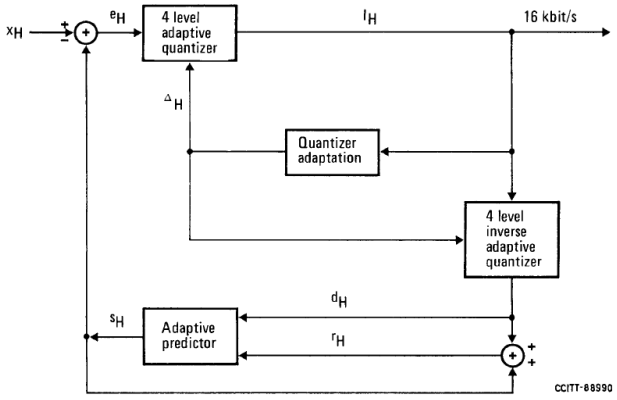


FIGURE V: HIGHER SUBBAND ENCODER

C. Multiplexer and demultiplexer

The multiplexer at the encoder is used to combine the two encoded signals from the lower and higher subband. If this is done the encoding process is completed, and an output signal of 64 kpbs is generated. At the decoder this signal is demultiplexed, such that the lower and higher subband can be decoded.

V. COMPARISON G.722 AND PHILIPS SUBBAND CODER

When comparing the structures of G.722 and the Philips Subband Coders, some similarities can be found. Both codecs work with subbands. In order to split the input signal into these subbands similar filters are used. Both codecs use QMF filters. Apart from this similarity the codecs differ greatly.

First of all the G.722 codec uses only 2 subbands, while the Philips subband coder uses 4 or 8 subbands. In the G.722 codec 75% of the available bits are assigned to the lower subband. This is because G.722 is focused on speech. Since there are almost no bits available for higher frequencies, this codec will not perform well for high frequency signals. The Philips Subband coder doesn't have this problem, because bits are assigned using the SFI. So every subband can get enough bits, even the subbands which contain the higher frequencies.

A second major difference is that G.722 uses ADPCM encoders, while the Philips subband coder uses an APCM encoder. Here the G.722 codec has an advantage because it uses prediction. However this makes the codec slightly more complex. But in our application this isn't a problem because the G.722 codec is implemented in hardware.

If we combine these facts, than in theory the Philips Subband Coder should perform better than the G.722 codec for music signals.

VI. PHILIPS SUBBAND CODER IMPLEMENTATION

To test the Philips Subband Coder one development board with two DSPs is used. One DSP is a CoolFlux DSP (NxH1210) the other chip is an NxH2180. The NxH2180 can also be used to connect two development boards wirelessly via magnetic induction. In this setup each development board represents a hearing aid. Since only the quality of the Philips Subband Coder needs to be examined, only one development board is used. Figure 6 shows the block diagram of the test setup.

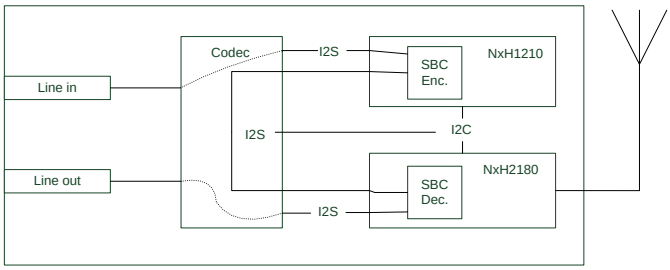


FIGURE VI: BLOCK DIAGRAM DEVELOPMENT BOARD

In this diagram three important components can be distinguished. The Codec is an ADC/DAC, it's is used to convert the analog signal to a digital signal and vice versa. The NxH1210 will encode the audio, the NxH2180 will decode the audio. So the audio comes from the line in and goes to the codec. Then it goes through the NxH1210 to be encoded. After that the encoded signal is sent to the NxH2180 and is decoded. In the final stage it is sent back to the codec, then to the line out.

The Philips Subband Encoder is programmed such that it's easy to test different configurations of the Philips Subband Coder. A number of different parameters can be set: the number of subbands (4 or 8), the block size (4,8,12,16) and the bitpool size. Other than that it is also possible to select in which way the audio is encoded. Four choices are available:

- Mono: only the left or right channel is encoded;
- Dual or stereo: these modes are quite similar both the left and right channel are encoded;
- Joint stereo: when this is selected left and right channel are encoded. But information that is the same in both channels is encoded only once, so this should get the best results.

In this setup with one development board the bitrate is limited to the bitrate of I²S, this is the bus used to transfer the audio samples. The maximum bitrate is 1024 kpbs for I²S in this setup, this value comes from 32 kHz sampling rate and 16 bit words, however when the Philips Subband Coder will be implemented using two development boards, the bitrate is limited to 166 kpbs. This limitation comes from the capacity of the wireless channel. For this reason the maximum bitrate is set to 166 kpbs in the setup with one development board.

In a first phase different configurations for the Joint, Stereo and Dual mode will be tested. When this is done the best configuration for each mode is selected. Then another test

is done by comparing all the selected configurations, in this phase the mono mode is also included. The listening test's are done using the MUSHRA (Multiple Stimuli with Hidden Reference and Anchor) test [10].

VII. MUSHRA LISTENING TEST

The MUSHRA listening test, is used for the subjective assessment of intermediate audio quality. This test is relatively simple. There are a few requirements for the test signals. They should not be longer than 20 s to avoid fatiguing of listeners. A set of signals to be evaluated by the listener consists of the signals under test, at least one anchor (in this test two anchors) and a hidden original signal. The listener can also play the original signal.

The anchors and the hidden reference are used to see if the results of a listener can be trusted. In this way more anomalies may be detected. The anchors are the original signal with a limited bandwidth of 3.5 kHz and 7 kHz. This is the original signal sent through low pass filters.

In the first phase 11 different audio signals are encoded with three different configurations for the three modes. These configurations can be found in table 2. So in this phase the listener is presented six signals to evaluate. This test is done for each mode, except for mono.

TABLE II: SBC CONFIGURATION PARAMETERS

	subbands	block size	bitpool	bitrate
Joint 1	8	16	35	166
Joint 2	4	16	16	164
Joint 3	8	8	28	164
Stereo 1	8	16	35	164
Stereo 2	4	16	16	160
Stereo 3	8	8	29	164
Dual 1	4	16	8	160
Dual 2	8	16	18	168
Dual 3	8	8	15	168

Then the best configuration is selected for each mode and a new test is done. Now the listener is presented seven signals to evaluate because now a mono configuration is also included.

The listener has to grade each signal between 0 and 5 (unacceptable to perfect). The grading allows steps of 0.1, so that enough scores are available.

Because the development boards are made for testing purposes, some noise is introduced to the audio output of these boards. Also the cables connecting the boards to the PC introduce noise. Therefore it was decided to generate the audio signals using a software encoder and decoder on a computer. This way no additional noise can occur and more accurate results are acquired. The noise introduced made it too easy to differentiate the original form the coded samples.

VIII. RESULTS

A. Phase 1

Table 3 gives the scores for the different configurations of the different modes. These values are the average score of 11 different audio signals.

TABLE III: SBC CONFIGURATION PARAMETERS

	conf1	conf2	conf3	Org.	anchor1	anchor2
joint	4,56	3,87	4,35	5,00	3,03	4,30
stereo	4,04	3,44	3,85	5,00	2,81	4,80
dual	3,69	4,38	4,35	5,00	3,10	4,86

B. Phase 2

Table 4 gives the results from the listening test with the different modes, these results are also the average of 11audio signals.

TABLE IV: SBC CONFIGURATION PARAMETERS

Joint1	Stereo1	Dual2	mono	Org.	anchor1	anchor 2
3,67	3,38	3,47	3,43	4,77	1,97	4,03

IX. DISCUSSION OF RESULTS

After examining the results of phase 1 the conclusion can be made that the configurations with 8 subbands and block length 16 always give the best results at a limited bitrate of 166 kpbs. In phase 2 the results show that the joint stereo mode is best. But the audio quality isn't very high. Artifacts can be heard, which is due to the limited bandwidth of 166 kpbs. The artifacts aren't audible when the frequency band is limited. In modern music though the frequency band is very wide, and this causes more artifacts.

In [4] the G.722 codec is evaluated. From these results it was concluded that for music signals a number of audible distortions were revealed that do not occur for speech signals. Also the perceived bandwidth of the coded music was less than 7 kHz. This is something that wasn't noticed during the listening tests of the Philips Subband Coder. The evaluation of G.722 also showed that more noise presented itself in the higher subband.

X. CONCLUSION

The main question in this paper, was if and how it is possible to improve audio quality for a hearing aid. This hearing aid was using a speech codec G.722. To improve quality the Philips Subband Coder is proposed. After looking at the structure of both codecs it can be concluded that the Philips Subband Coder performs better for music signals than G.722. But at the moment there is a limitation to a bitrate of 166 kpbs. For this reason artifacts are heared when using the Philips Subband Coder, although when compared with G.722 the sound itself is better. With G.722 the higher frequencies don't really come through, the Philips Subband Coder solves this problem. When new hardware is available, which allows higher bitrates, the Philips Subband Coder is a possible choice for this application. Most important reasons for this are its low

complexity, thus low memory and MIPS requirements. Also, this codec has a low delay making it ideal for hearing aids.

ACKNOWLEDGEMENTS

I thank Steven Daenen for giving me the chance for doing this research at NXP, also I would like to thank Koen Derom for his help at NXP. Further I want to thank Paul Leroux guiding me through this project.

REFERENCES

[1] Hawley, M. L., Litovsky, R. Y., and Culling, J. F., ‘‘The benefit of binaural hearing in a cocktail party: Effect of location and type of interferer,’’ J. Acoust. Soc. Am. **115**, 2004, pp. 833–843.

[2] F. de Bont, M. Groenewegen, and W. Oomen, “A High Quality Audio-Coding System at 128kb/s,” in Proceedings of the 98th AES Convention, Paris, France, Feb. 1995.

[3] M. Lutzky, G. Schuller, M. Gayer, U. Krämer, and S. Wabnik, “A guideline to audio codec delay,” in Proceedings of the 116th AES Convention, Berlin, Germany, May 2004.

[4] S.M.F. Smyth et al., “An independent evaluation of the performance of the CCITT G.722 wideband coding recommendation”. IEEE Proc. ICCASP, 1988, pp 2544-2547.

[5] “Advanced audio distribution profile (A2DP) specification version 1.2,” http://www.bluetooth.org/, Apr. 2007, bluetooth Special Interest Group, Audio VideoWG.

[6] P.P. Vaidyanathan, "Quadrature Mirror Filter Banks, M-Band Extensions and Perfect-Reconstruction Techniques", IEEE ASSP magazine, July 1987, pp. 4 - 20.

[7] J.H. Rothweiler, “Polyphase quadrature filters: A new subband coding technique”, IEEE Proc. ICCASP, 1983, pp 1280-1283.

[8] ITU Recommendation G.722 , “ 7 kHz audio-coding within 64 kbit/s.”, November 1988

[9] P. Mermelstein, “G.722, a new CCITT coding standard for digital transmission of wideband audio signals”, *IEEE Communication Magazine*, vol. 26, February 1988, pp. 8-15.

[10] ITU-R, “Method for the subjective assessment of intermediate quality levels of coding systems,” Recommendation BS.1534-1, Jan. 2003.

Performance and capacity testing on a Windows Server 2003 Terminal Server

Robby Wielockx
K.H. Kempen, Geel
robby.wielockx@gmail.com

Rudi Swennen
TBP Electronics, Geel
rswennen@tbp.eu

Vic Van Roie
K.H. Kempen, Geel
vic.van.roie@khk.be

***Abstract*—Using a Terminal Server instead of just a traditional desktop environment has many advantages. This paper illustrates the difference between using one of those regular workstations and using a virtual desktop on a Terminal Server by setting up an RDC session. Performance testing indicates that the Terminal Server environment is 24% faster and handles resources better.**

We have also done capacity testing on the Terminal Server, which results in the number of users that can connect to the server at the same time and what can be done to increase this. The company this research has been conducted for, desired forty concurrent terminal users. Unfortunately, our results turned out that at this moment only seven users can be supported, without extending existing hardware (memory and CPU).

I. Introduction

Windows Server 2003 has a Terminal Server component which allows a user to access applications and desktops on a remote computer over a network. The user works on a client device, which can be a Windows, Macintosh or Linux workstation. The software on this workstation that allows the user to connect to a server running Terminal Services is called Remote Desktop Connection (RDC), formerly called Terminal Services Client. The RDC presents the desktop interface of the remote system as if it were accessed locally.

In some environments, workstations are configured so that users can access some applications locally on their own computer and some remotely from the Terminal Server. In other environments, the administrators choose to configure the client workstations to access all of their applications via a Terminal Server. This has the advantage that management is centralized which makes it easier to do. These environments are called Server-Based Computing.

The Terminal Server environment used for performance and capacity testing as described in this paper are Server-Based Computing environments. The Terminal Server is accessed via an RDC and the Terminal Server delivers a full desktop experience to the client. The Windows Server 2003 environment uses a specially-modified kernel which allows many users to connect to the server simultaneously. Each user is running its own unique virtual desktop and is not influenced by actions from other users. A single server can support tens or even hundreds of users. The number of users a Terminal Server can support depends on which applications they use and of course it depends strongly on the server hardware and the network configuration. Capacity testing determines this number of users and also possible bottlenecks in the environment. By upgrading or changing server or network hardware, these bottlenecks can be lifted and the server is able to support more users simultaneously.

This research is done for a company which has eighty Terminal Server User CALs (Client Access Licenses). Each CAL enables one user to connect to a Terminal Server. At the moment, the company has two Terminal Servers available so ideally they would like each server to support forty users. By testing the capacity of each Terminal Server we can determine the number of users each server can support and discover which upgrades can be done to raise this number to the desired level. A second part is testing the performance of working with a Terminal Server compared to working without a Terminal Server and just a workstation for each user (which is the current way of working in the company).

II. Performance testing

A. Intention

The purpose of the performance testing is to compare the use of a traditional desktop solution with the Terminal Server solution which provides a virtual desktop. We want to examine if users experience a difference between the two solutions in the field of working speed, load times and overall easiness of use. To do this, a user manually performs a series of predefined tasks on both the desktop and the virtual desktop. For the users, the most important factor is the overall speed of the task. This speed will be different at both tests because the speed of opening programs and loading documents on two different machines is never the same.

B. Collecting data

1) Series of user actions: The series of actions that a user has to perform during this performance testing consists of three parts. The user needs to execute these actions at a normal working speed, one after another. To eliminate errors as a result of hazards, the series of actions are performed multiple times on both desktops. We than take the average of these results to draw

the conclusions. First, the user opens the program Isah and performs some actions. Next, the user opens Valor Universal ¡viewer and loads a PCB data model. Thereafter, the user opens Paperless, which is an Oracle database, and loads some documents. Finally, the user closes all documents and programs, after which the test ends.

2) Logging data: During the execution of the actions, data has to be logged. This can be done in two ways: by using a third-party performance monitoring tool or by using the Windows Performance MMC (Microsoft Management Console) snap-in. The first way offers more enhanced analysis capabilities, but is also more expensive. For this reason, we use the MMC which has sufficient features in our situation. In the MMC we can add performance counters that log to a file during the test. After the test, the file can be imported into Microsoft Excel to be examined. For this performance test, we need to choose counters to examine the speed of the process and the network usage. These are the most important factors. Therefore the counters we add are:

- Process > Working Set > _Total
- Memory > Pages Output/sec
- Network Interface > Bytes Total/sec
- Network Interface > Output Queue Length

By default, the system records a sample of data every fifteen seconds. Depending on hard disk space and test size, this sample frequency can be increased or decreased. Because the test endures only a few minutes, we choose a sample frequency of just one second.

3) Specifications: The traditional workstation has an Intel Core2 CPU, 2.13 GHz and 1.99 GB of RAM. The installed operating system is Microsoft Windows XP Professional, v. 2002 with Service Pack 3. Its network card is a Broadcom NetXtreme Gigabit Ethernet card. The Terminal Server has an Intel Xeon CPU, 2.27 GHz and 3 GB of RAM. The operating system is Microsoft Windows Server 2003 R2 Standard Edition with Service Pack 2. It has an Intel PRO_1000 MT network card.

C. Discussion

1) Speed: The most important factor is obviously the execution speed of the test. When performing the actions on the traditional desktop, it takes an average of 198 seconds to perform all predefined tasks. On the Terminal Server on the other hand, it only takes an average of 150 seconds. This means that in this case the Terminal Server desktop environment is 48 seconds or approximately 24% faster than the regular desktop. Saving almost a minute of time when performing a series of tasks that takes only about 3.5 minutes is a lot.

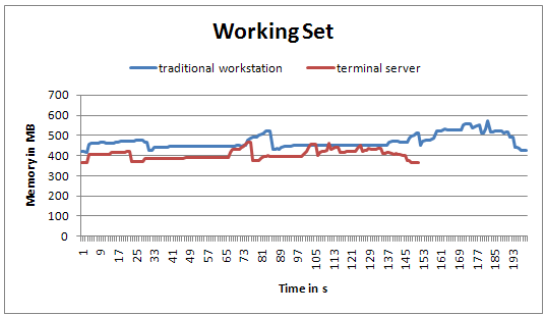


Fig. 1. Output from the Process > Working Set > _Total counter

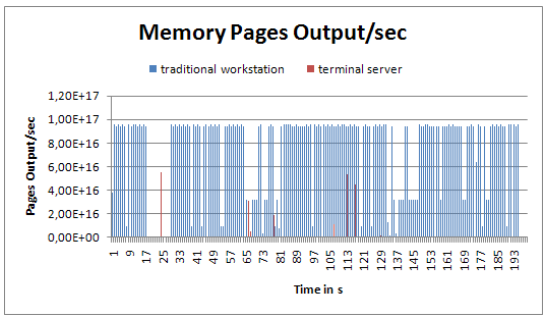


Fig. 2. Output from the Memory > Pages Output/sec counter

2) Memory: Figure 1 shows the output from the working set counter. This counter shows the total of all working sets of all processes on the system, not including the base memory of the system, in bytes. First of all, the figure also shows the difference in execution speed we discussed in II-C1. We can see that for the same series of actions, it takes significantly less time to perform then on the Terminal Server desktop.

Another conclusion that this data shows is the memory usage. When executing tasks on the regular desktop, the memory usage varies between 400 MB and 600 MB, whereas the memory usage in the virtual desktop environment varies only between 350 MB and 450 MB. We can conclude that the virtual desktop uses slightly less memory than the regular desktop and the variations are smaller.

The output from the Pages Output/sec counter is shown in figure 2 and indicates how many times per second the system trims the working set of a process by writing some memory to the disk in order to make physical memory free for another process. This is a waste of valuable processor time, so the less the memory has to be written to the disk, the better. Windows doesn't pay much attention to the working set when physical memory is plentiful: it doesn't trim the working set by writing unused pages to the hard disk. In this case, the output of the counter is very low. When the physical memory utilization gets higher, Windows will start to trim the working set. The output from the Pages Output/sec counter is much higher.

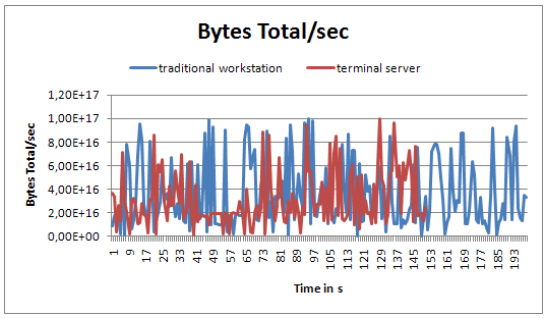


Fig. 3. Output from the Network Interface > Bytes Total/sec counter

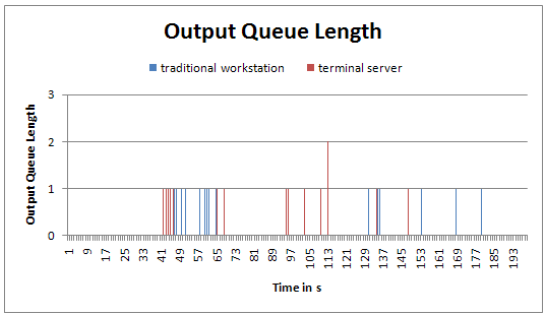


Fig. 4. Output from the Network Interface > Output Queue Length counter

We can see in figure 2 that there is plenty memory on the Terminal Server. There is no need to trim the inactive pages. On the other hand, when performing the actions on a regular desktop, a lot of pages need to be trimmed to make more physical memory free, which results in more unwanted processor utilization and thus a longer overall speed. The above explanation indicates that the working set of the Terminal Server environment in figure 1 isn't a good representation compared to the working set of the traditional desktop: it shows active and inactive pages, whereas the traditional desktop output shows mostly active pages.

3) Network: Also important when considering performance is the network usage. The output from the Network Interface Bytes Total/sec is shown in figure 3. The figure indicates that there is slightly more network traffic when working with the regular desktop environment. The reason for this is that the desktop has to communicate with the file servers of the company, which are in the basement in the server room. The virtual desktop on the Terminal Server also has to communicate with these file servers, but the Terminal Server itself is also located in the server room, which means the distance to cross is much smaller. Also, the speed of the network between the two servers (1 Gbps) is greater than the speed of a link between a regular workstation and the servers in the server room (100 Mbps).

Figure 4 shows the output from the Network Interface Output Queue Length counter.If this counter should have a sustained value of more than two, then performance could be increased by for example replacing the network card with a faster one. In our case when testing the network performance between a regular workstation and a virtual desktop on a Terminal Server, we see that both the desktop as the Terminal Server suffice. But we have to keep in mind that during the testing, only one user was active on the Terminal Server. The purpose of the Terminal Server is to provide a workspace for multiple users, so the output from the Queue Length counter will be higher.

4) User experience: Also important is how the user experiences both solutions. The first solution, which is using a regular desktop, is familiar for the user. The second solution, which is accessing a virtual desktop on a Terminal Server by setting up an RDC connection, is not so familiar to most normal users. Most of them havent used RDC connections before and having to cope with a local desktop and on top of that a virtual desktop can be confusing. This problem can be solved by setting up the RDC session automatically when the client computer is starting up, which eliminates the local desktop and leaves only one virtual desktop, which is practically the same for an unexperienced user. The only difference they experience is that most virtual desktop environments are heavily locked down, to prevent users from doing things on the Terminal Server theyre not supposed to.

D. Results

We have tested the performance of both solutions by performing the same series of actions on the traditional desktop and the virtual desktop. The testing indicates that the Terminal Server environment is 24% faster than the regular environment. It also scores better regarding memory and network usage. Working with a Terminal Server environment has many advantages, but definitely saving time is an important one.

III. Capacity testing

A. Intention

Now that we know the difference between the traditional desktop solution and the Terminal Server virtual desktop solution, we need to know how many users the Terminal Server can support. This number can vary greatly because of different environments, network speed, protocols, Windows profiles and hardware and software revisions. For this testing, we use a script for simulating user load on the server. Instead of asking real users to use the system while observing the performance, a script simulates users using the system. Using a script also gives an advantage: you can get consistent, repeatable loads.

The approach behind this capacity testing is the following. First, we did the test with just one user connected to the Terminal Server. The script runs, simulates user activity and the performance is monitored. Next, we added one user and repeated the test. Thereafter we did the test with three and four users, because we only had four machines at our disposal. Afterwards, the results from the four tests can be compared.

B. Simulating user load

First, we determined the actions and applications that had to be simulated. We used the same series of user actions as in section II-B1. To simulate a normal user speed and response time, we added pauses in the script. The program we used for creating a script is AutoIt v32[1]. AutoIt is a freeware scripting language designed for automating the Windows GUI. It uses simulating keystrokes, mouse movements and window and control manipulation to automate tasks. When the script is completed, you end with a .exe file that can be launched from the command line. When the script is launched, it takes over the computer and simulates user activity.

C. Monitorring and testing

1) Performance monitoring: During the testing process, the performance has to be monitored. For collecting the data, I use the Windows Performance MMC, which I also used for logging the data when testing the performance (see section II-B2). For testing the capacity, it is important to look at how the Terminal Server uses memory. Other factors to be examined are the execution speed, the processor and the network usage. The counters we added in the Windows Performance MMC to examine the testing results are the following:

- Process > Working Set > _Total
- Memory > Pages Output/sec
- Network Interface > Bytes Total/sec
- Network Interface > Output Queue Length
- Processor > % Processor Time > _Total
- System > Processor Queue Length

The first four counters were also added when testing the performance.

2) Testing process: When the script is ready and the monitoring counters are set up correctly, the actual testing process can begin. When testing with tens of users, the easiest way to do this is by placing a shortcut to the test script in de Startup folder so that the script runs when the RDC session is launched. Because the testing in our case is only with four different users, we manually launch the script in each session. For testing, we could use four different workstations. On each workstation, we launched one RDC session to the Terminal Server. At approximately the same moment, we kicked-off the simulating script.

[1]http://www.autoitscript.com/autoit3/index.shtml

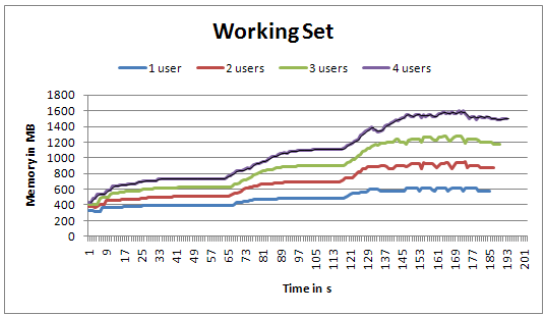


Fig. 5. Output from the Process > Working Set > _Total counter

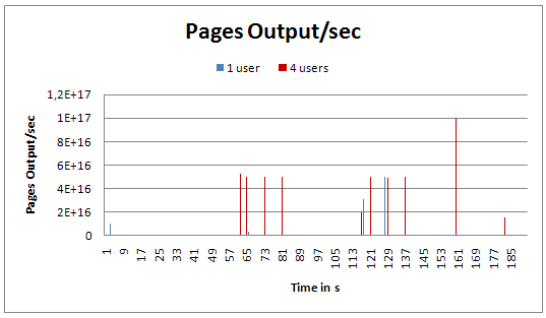


Fig. 6. Output from the Memory > Pages Output/sec counter

Having more RDC sessions on a single workstation is possible, but in this case wasnt usable. Because the script simulates mouse movements and keystrokes, it only works at one RDC session at the time per workstation. When having multiple sessions on a single workstation, only the active session - the session at the front of the screen - would run the script correctly. The session of which the window is minimized or behind another RDC session window would not execute the script correctly. Therefore, because we had four machines at our disposal, we could only run four RDC sessions which could run the script correctly at the same time.

D. Discussion

1) Memory: Figure 5 shows the output from the Working Set counter, which is the total of all working sets of all processes on the system in bytes. This number does not include the base memory of the system. The first thing we can conclude is that the execution time does not increase significantly when adding more users to the server (around 2 seconds per extra user).

Next, we can look at the memory usage. One user running the simulation script uses a maximum of around 600 MB. We see that for each extra user who runs the script, the memory usage raises with approximately 350 MB. For example, when three users are running the script, the Working Set counter has a maximum of 1300 MB (600 MB for one user and 2 times 350 MB for the extra two users). Normally we would expect the memory used when three users are running the script to

be 1800 MB (600 MB times 3), when in fact it turns out to be only 1300 MB.

The reason for this is that a Windows Server 2003 Terminal Server uses the memory in a special way. For example, when ten users are all using the same application on the server, the system does not need to physically load the executable of this application in the memory ten times. It loads the executable just one time and the other sessions are referred to this executable. Each session thinks that they have a copy of the executable in their own memory space, which is obviously not true. This way, the operating system can save memory space and the overall memory usage is lower.

The Terminal Server has 3 GB of RAM (see section II-B3). We can calculate the maximum number of users the server could handle with the following equation:

$$600 + (x - 1) * 350 \leq 3000 \quad (1)$$

$$x \leq 7,86 \quad (2)$$

Only seven users can use the Terminal Server at one time, when performing the same actions as simulated by the script. This is a lot less than the desired number of forty. If every user should perform in this way, the memory of the server should be increased to 14 GB (see the equation below).

$$600 + (40 - 1) * 350 = 14000 \quad (3)$$

The output from the Pages Output/sec counter is shown in figure 6. This counter indicates how many times per second the system trims the working set of a process by writing some memory to the disk in order to make physical memory free for another process. When the system is running low on physical memory when more users are connected to the Terminal Server, the Pages Output/sec counter will start to show high spikes. Then the spikes will become less and less pronounced until the counter begins rising overall. The point where spiking is finished and the overall rising begins is a critical point for the Terminal Server. This indicates that the Terminal Server hasnt enough memory and could benefit from more memory. If this counter does not have an overall rise after the spiking is finished, then this indicates that the server does have enough memory.

As described in section II-C2, the system only trims memory when physical memory utilization gets higher. We can see in the figure that the counter values are low, even when four users are running the script. This means that inactive pages aren't trimmed and are still in the working set. Therefore we can conclude that more than seven users could use the Terminal Server at one time (although the exact number can't be determined from the results).

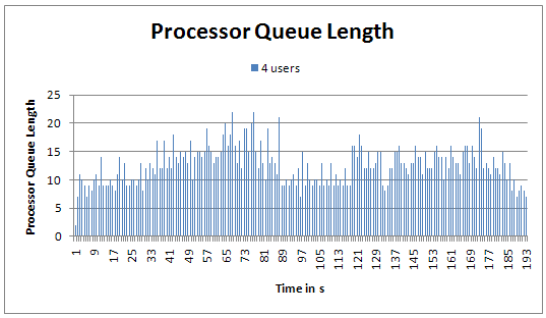


Fig. 7. Output from the System > Processor Queue Length

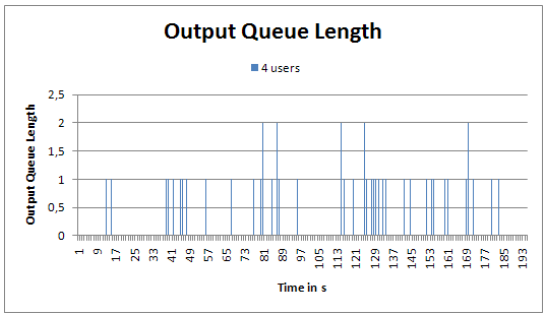


Fig. 8. Output from the Network Interface > Output Queue Length counter

Note, the actions performed in this test are extreme and probably most users never will access al programs or load all documents at the same time. When studying two real users working at the Terminal Server during their job, memory usage for both employees ranges from 90 MB to 160 MB. This means that the real users use less memory than the simulation script. Therefore the Terminal server can support more users than the calculated number of 7.

2) Processor: The output from the Processor Time counter indicates that there isn't a sustained value of 100% utilization, which should mean that the processors aren't too busy.

However, when we look at figure 7, which shows the output from the Processor Queue Length counter, we can see that there is a sustained value of around 10 with peaks up to 20. The Queue Length counter indicates the number of requests which are backup up as they wait for the processors. If the processors are too busy, the queue will start to fill up quickly, which indicates that the processors aren't fast enough. The queue shouldn't have a sustained value of 2, which is the threshold. Figure 7 show that the counter has a sustained value significantly greater than 2, so the processors of the Terminal Server aren't fast enough. This will probably result in a decrease of performance when more users are using the server. This can be resolved by upgrading the processors.

3) Network: Network usage can be a limiting factor when it comes to Terminal Server environments. It is the interface between the Terminal Server and the network file servers that normally cause the blockage, not the RDC sessions as one would think. The sessions itself dont require a lot of network bandwidth, depending on which settings are configured for the RDC session (think about themes, desktop background, color depth, ...). For our Terminal Server environment, the network isnt likely to be a limiting factor. Should it have been one, then fixing this bottleneck is very easy. You just have to put a faster NIC in the server or implement NIC teaming or full duplexing to double the interface of the server.

Just like the Processor Queue Length which indicates whether or not the processor is limiting the number of user sessions on the Terminal Server (see section III-D2), there is a Network Interface Output Queue Length which indicates whether or not the network is the bottleneck. The output from the counter which indicates this queue length is shown in figure 8. If the value of the counter sustains more than two, then action should be taken if we want more users on our Terminal Server. In our testing environment with one user RDC session, the counter reaches three times the value of two and when testing with four users, the counter indicates a few times the value of three. Because this value isnt sustained, there is no problem with our network interface and therefore the network isnt the limiting factor.

E. Results

We have tested the capacity of the Terminal Server by comparing the results from one RDC session running a script with the results from multiple RDC sessions running the same script simultaneously. Most likely in the company environment with the current server hardware, memory is the bottleneck when it comes to server capacity. The testing indicates that the Terminal Server could support around 7 users, in the most extreme conditions of our script. The goal for the company is to support forty users per Terminal Server, so upgrading server memory is inevitable. Also the processors need to be upgraded.

IV. Conclusion

There are differences between using a traditional workstation and using a virtual desktop environment on a Terminal Server, which can be accessed by setting up an RDC session between a client machine and the Terminal Server itself. By testing the performance, we can examine these differences in the field of working speed, load times and overall easiness of use. To compare these two solutions, we needed to collect the data. First, we manually performed a series of user actions on a traditional workstation and logged certain counters. Afterwards, we manually performed the same series of actions on a virtual desktop on the Terminal Server. By comparing the results we have learned that first of all the Terminal Server environment executes the same series of actions 24% faster than de traditional workstation. We also concluded that memory usage and network usage is more efficient in a Terminal Server environment. It is also pointed out that, out of user experience, the traditional workstation is more familiar and easier to cope with than the Terminal Server environment with a local desktop and on top of that a virtual, remote desktop.

Next, it is important to know the capacity of your Terminal Server. This is indicated by the number of users that can access and use the Terminal Server simultaneously. This is tested by comparing a predefined series of actions executed in only one user session with the same predefined series of actions in two, three and four different user sessions. The user actions were simulated by using a script. We learn that the Terminal Server in our environment with the current server hardware and 3 GB of RAM can only support 7 users. When considering real users, the conditions are less extreme and the server can probably support a lot more users. Adding more memory results in more users. Other bottlenecks in Terminal Server environments are processor time and network usage. Processor time in our case is likely to be a bottleneck depending on the Processor Queue. Also the network isnt the limiting factor and if it ever turns out to be one, installing a faster NIC in the server fixes this factor in an easy way.

V. Acknowledgements

The authors would like to thank the ICT team from TBP Electronics Belgium, situated in Geel, for help and support. Special thanks to ICT team manager Rudi Swennen.

References

[1] B.S. Madden and R. Oglesby, *Terminal Services for Microsoft Windows Server 2003: Advanced Technical Design Guide*, 1st-ed. Washington DC, USA: BrianMadden.com Publishing, 2004.

[2] E. Sheesley., *SolutionBase: Working with Microsoft Windows Server 2003's Performance Monitor*, TechRepublic.com, 2004.

[3] A. Silberschatz, P.B. Galvin, G. Gagne, *Operating System Concepts*, 8th-ed. Asia: John Wiley & Sons Pte Ltd, 2008.

[4] R. Morimoto, A. Abbate, E. Kovach and E. Roberts, *Microsoft Windows Server 2003 Insider Solutions*, 1st-ed. USA: Sams Publishing, 2004.

[5] D. Bird, "Keep Tabs on Your Network Traffic". Available at http://www.enterprisenetworkingplanet.com/netsysm/article.php/10954_3328281_1, February 2010.

[6] "Terminal Server Capacity Planning". Available at http://technet.microsoft.com/en-us/librarycc751284.aspx, February 2010.

[7] "What is that Page File for anyway?". Available at http://blogs.technet.com/askperf/archive/2007/12/14/what-is-the-page-file-for-anyway.aspx, February 2010.

[8] "AutoIt Documentation". Available at http://www.autoitscript.com/autoit3/docs/, February 2010.

Silverlight 3.0 application with a Model-View-Controller designpattern and multi-touch capabilities.

Geert Wouters

[geert.wouters@edu.khk.be]

IBW, K.H. Kempen (Associatie KULeuven)
Kleinhoefstraat 4
B-2440 Geel (Belgium)

***Abstract*—The technology and the availability of multi-touch devices is rapidly growing. Not only the industry is making these devices but also several groups of enthusiasts that are making their own home-made multi-touch table like the "Natural User Interface group". One of the methods they use is Frustrated Total Internal Reflection (FTIR) which was used for testing. To use these devices efficiently, it is necessary that new technologies are being introduced. Many of the software technologies that are used nowadays are not able to communicate with multi-touch devices or gestures that are made on these devices. So, a multi-touch table that communicates with Silverlight 3.0 (released in July 2009) will be presented. This programming language supports multi-touch but it doesn't recognize any gesture. A complete description of the most intuitive gestures and how to integrate them into a Silverlight 3.0 application will be discussed. We will also describe how to connect this application with a database to build a secure and reliable B2B, B2C or media application.**

I. Introduction and related work

For testing the multi-touch capabilities of a Silverlight 3.0 application we used the multi-touch table that was made in a previous work [1] by Nick Van den Vonder and Dennis De Quint. This multi-touch table was based on a research by Jefferson Y. Han [2]. The multi-touch screen uses FTIR to detect fingers, also called "blobs", that are pressed on the screen. On Figure 1 we see how FTIR can be used with a webcam that only captures infrared light by using an infrared filter. This infrared light is generated by the LED lights that are send through the acrylic pane. If you put a finger on the screen the infrared light will be sent to the webcam. The webcam captures this light and will be sent to the connected computer. You can also notice on Figure 1 that a projector is used. This is not really necessary because the sensor (webcam) can be used standalone. Without a projector the multi-touch table is completely transparent and therefore it is particularly suited for use in combination with rear-projection. On the rear side of the waveguide a diffuser (e.g. Rosco gray) is placed which doesn't frustrate the total internal reflection because there is a tiny gap of air between the diffuser and the waveguide. The diffuser doesn't affect the infrared image that is seen by the webcam, because it is very close to the light sources (e.g. fingers) that are captured.

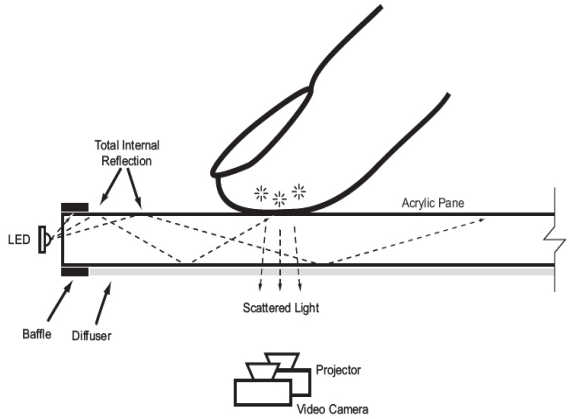


Figure 1: Schematic overview of a home-made multi-touch screen. [2]

Why multi-touch?
The question is why we would use multi-touch technology. The problem lies in the classic way to communicate with a desktop computer. Mostly we use indirect devices with only one point of input such as a mouse or keyboard to control the computer. With the multi-touch technology there will be a new way to human computer interaction because these devices are capable to track multiple points of input instead of only one point. This property is extremely useful for a team collaborating on the same project or computer. It gives a more natural and intuitive way to communicate with the team members.

II. Silverlight 3.0

Now that we have the hardware to test the multi-touch capabilities we need the appropriate software to communicate with the multi-touch device. In the company, Item Solutions, where the research was made, they introduced us to the programming language Microsoft Silverlight 3.0. Silverlight 3.0 is a cross-over browser plugin which is compatible with multiple web browsers on multiple operating systems e.g. Microsoft Windows and Mac OS X. Linux, FreeBSD and other open source platforms can use Silverlight 3.0 by using a free software implementation named Moonlight that is developed by Novell in cooperation with Microsoft. Mobile devices, starting with Windows Mobile and Symbian (Series 60) phones, will likely be supported in 2010. The Silverlight 3.0 plugin (± 5MB) includes a subset of the .NET framework (± 50MB). The main difference between the full .NET framework and the subset of Silverlight 3.0 is the code to connect with a database. Silverlight 3.0 works client-side and can not directly connect to a database. For the connection it has to use a service-oriented model that can communicate across the web like Windows Communication Foundation (WCF).

Windows Communication Foundation is a new infrastructure for communication and is an extention of the expanded set of existing mechanisms such as Web services. Windows Communication Foundation is a new infrastructure for communication and is an extention of the expanded set of existing mechanisms such as Web services. WCF makes it possible for developers using a simple programming model to build safe, reliable and configurable applications. This means that WCF provides a robust and reliable communication between client and server.

Not only the connection with the database can create a qualitybased application. It is also necessary that a good structure for the code is used.

For this research the Model-View-Controller designpattern is used. This pattern splits the design of complex applications into three main sections each with their own responsibilities:
Model: A model manages one or more data elements and includes the domain logic. When a data element in the model changes, it notifies its associated views so they can refresh.
View: A view renders the model into a form that is suitable for interaction what typically results in a user interface element.
Controller: A controller receives input for the database through WCF and initiates a response by making calls to the model.

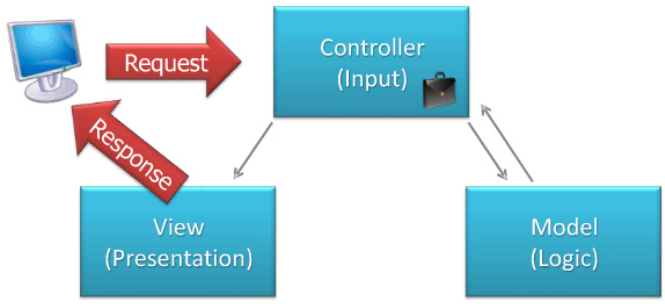


Figure 2: Model-View-Controller model. [3]

The advantages of using a designpattern is that the readability and reusability of the code significantly increases and it is designed to solve common design problems.

Silverlight 3.0 is not only capable to use these two concepts but there is also a minimal support for multi-touch capabilities. The only thing that Silverlight 3.0 can detect is a down, move and up event for a blob/touchpoint (point or area that is detected).

III. Multi-touch gestures

The paper “User-Defined Gestures for Surface Computing” [4] by J. O. Wobbrock, M. R. Morris and A. D. Wilson researched the behaviour how people want to interact with a multi-touch screen. In total they analyzed 1080 gestures from 20 participants for 27 commands performed with 1 or 2 hands. The gestures we needed and implemented where “Single select: tap”, “Select group: hold and tap”, “Move: drag”, “Pan: drag hand”, “Enlarge (Shrink): pull apart with hands”, “Enlarge (Shrink): pull apart with fingers”, “Enlarge (Shrink): pinch”, “Enlarge (Shrink): splay fingers”, “Zoom in (Zoom out): pull apart with hands”, “Open: double tap”.

Single select: tap
For a “single select: tap” of an object, see Figure 3, it is necessary that we can detect where the user

pressed the multi-touch screen. These coordinates must be linked to the corresponding object. On this object we checked if there occurred a down and rapidly up event. If these two events occur in a single object the object must be selected.
In Silverlight 3.0 the code below can be used to select an object.

```
Touch.FrameReported += new
              TouchFrameEventHandler(
                  TP ActionReported);
TouchPointCollection tps
            = e.GetTouchPoints(null);
foreach (TP tp in tps){
  switch (tp.Action){
    case TouchAction.Down: ...
    case TouchAction.Move: ...
    case TouchAction.Up:   ...
  }
}
```

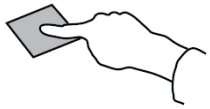


Figure 3: Single select tap. [4]

Select group: hold and tap
To select more than one object, see Figure 4, we can reuse Code 1 to select more objects at the same time. So here we have to detect multiple select tap events for multiple objects. Because there is no timer function in Silverlight 3.0, the code below can be used to make a hold function.

```
long timeInterval = 1000000;100ms
if ((DateTime.Now.Ticks - LastTick) <
                        timeInterval){
  selectedObject.Select();
}
LastTick = DateTime.Now.Ticks;
```

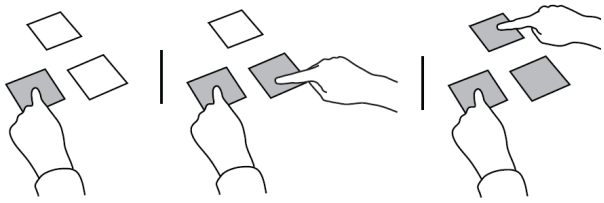


Figure 4: Select group: hold and tap. [4]

Move: drag
The move action, see Figure 5, can be realized by using the move event in Silverlight 3.0 of a blob. If a blob gives a down event followed by a move event, the object must be moved equal to the movement of the blob.
In Silverlight 3.0 we can simply change the position of elements to change the Left and Top property of the element.

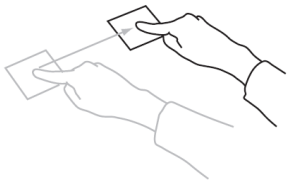


Figure 5: Move: drag. [4]

Pan: drag hand
For this gesture, see Figure 6, the method above can be reused, but now we first have to detect which blobs are in the object. From all the points in the object we have to calculate the midpoint by equation 1.

$$x, y = \left(\frac{x_1 + x_2 + \ldots + x_n}{n}\right), \left(\frac{y_1 + y_2 + \ldots + y_n}{n}\right) \quad (1)$$

When a blob moves, only the value of the moving blob has to change in equation 1. This results in a movement of the midpoint. Therefore the object has to move equal to the movement of the midpoint.
In Silverlight 3.0 we can use the code below to calculate the midpoint of all points.

```
foreach (KeyValuePair<int, Point>
origPoint in origPoints){
  totalOrigXPosition +=
                    origPoint.Value.X;
  totalOrigYPosition +=
                    origPoint.Value.Y;
}
double commonOriginalXPosition =
                  totalOrigXPosition /
                     origPoints.Count;
double commonOriginalYPosition =
                  totalOrigYPosition /
                     origPoints.Count;
Point commonOrigPoint = new
            Point(commonOrigXPosition,
                  commonOrigYPosition);
```

Figure 6: Pan: drag hand. [4]

Enlarge (Shrink)
When we speak about multi-touch most people think about the resizing or enlarging and shrinking of an object, see Figures 7 and 8, by using two points moving from or towards each other. If there are only two blobs in the object we can measure the distance of the two points by equation 2.

$$dist. = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2)$$

If there are more than two blobs in the objects we first need to calculate the midpoint by equation 1. We then have to determine the sum of the distances of all the points to the midpoint. So by every movement of a blob we only need to calculate the distance of the blob to the midpoint and change it with his previous value in the sum.
In Silverlight 3.0 we can use the code below to calculate the resize factor of all points. We have split the code into an x-component and a y-component. It is also possible to calculate the global resize factor with a little change.

```
totOrigXDist += Math.Sqrt(
   Math.Pow(commonOriginalPoint.X -
         originalPoint.Value.X, 2));
totOrigYDist += Math.Sqrt(
   Math.Pow(commonOriginalPoint.Y -
         originalPoint.Value.Y, 2));

selectedObject.Resize(((totNewXDist -
   totOrigXDist) / MTObject.Width) /
            (newPoints.Count / 2.0),
      ((totNewYDist - totOrigYDist) /
                  MTObject.Height) /
           (newPoints.Count / 2.0));
```

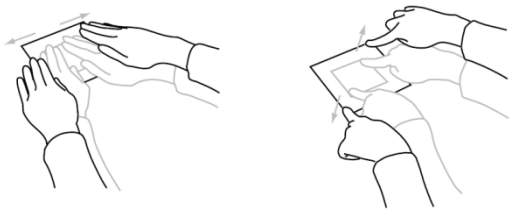


Figure 7: Enlarge (Shrink): pull apart with hands and fingers. [4]

Figure 8: Enlarge (Shrink): pinch and splay fingers. [4]

Zoom in (Zoom out)
The zoom in and zoom out function, see Figure 9, is very similar to the enlarge and shrink function explained before. The only difference is that the resize function is applied on the background or parent container of the object. This means that the resize factor of all the objects in the parent container needs to change depending on the resize factor.

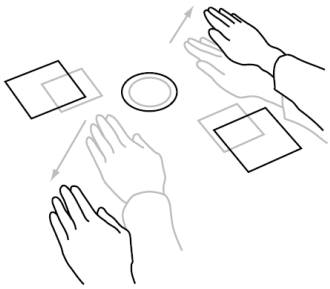


Figure 9: Zoom in (Zoom out). [4]

Open: double tap
This action, see Figure 10, can be detected by using rapidly two single select taps after each other. Because this is no standard gesture in Silverlight 3.0, we have to create this event manually. The key question of the double click event is his time-out. This must be carefully chosen so that the user has the best look and feel experience with the multi-touch application. According to MSDN, Windows uses a time-out of 500 ms (0,5 s). This time-out however, was too long to be useful in a multi-touch environment. It did not feel naturally. For instance, if you want to move an object from the top right corner to the bottom left corner, you normally use your right hand first to move it to the middle of the screen, then you use your left hand to move it from the middle to the left bottom corner. With a time-out of 500 ms it was not comfortable to wait while this time-out was expired. If the user however touches the object withing the time-out, the code of the doubleclick action will be executed what not always will be the intention of the user. From our multi-touch experience we took 250 ms as time-out. This gives a very intuitive feeling for this action. The code that can be used is already used for the hold function in section Select group: hold and tap. With a little modification the code will be useful in this context.

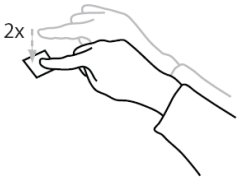


Figure 10: Open: double tap. [4]

IV. Conclusion

Silverlight 3.0 is a brand-new technology that is very promising for a multi-touch experience on desktop computers and in the future even mobile phones. The multi-touch support is not very extended but it is widely customisable. That makes it very useful for many programmers who are familiar with C#.NET and the .NET framework to

work with. As described, it is possible to implement many multi-touch gestures such as “Single select: tap”, “Select group: hold and tap”,“Move: drag”, “Pan: drag hand”, “Enlarge (Shrink): pull apart with hands”, “Enlarge (Shrink): pull apart with fingers”, “Enlarge (Shrink): pinch”, “Enlarge (Shrink): splay fingers”, “Zoom in (Zoom out): pull apart with hands”, “Open: double tap”. For accessing data it can easily make use of webservices like Windows Communication Foundation (WCF) for pulling data out of a database by using the secure and reliable Model-View-Controller (MVC) model.

REFERENCES

[1] N. Van den Vonder and D. De Quint, "Multi Touch Screen", Artesis Hogeschool Antwerpen, 2009, pp. 1-83.
[2] J. Y. Han, "Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection", Media Research Laboratory New York University, New York, 2005, pp. 115-118.
[3] M. Ballіauw, "ASP.NET MVC Wisdom", Realdolmen, Huizingen, 2009, pp. 1-13.
[4] J. O. Wobbrock, M. R. Morris and A. D. Wilson, "User-Defined Gestures for Surface Computing", Association for Computing Machinery , New York, 2009, pp. 1083-1092
[5] K. Dockx, "Microsoft Silverlight Roadshow Belgium", Realdolmen, Huizingen, 2009, pp. 1-21.

Comparative study of programming languages and communication methods for hardware testing of Cisco and Juniper Switches

Robin $Wuyts^1$, Kristof $Braeckman^2$, Staf $Vermeulen^1$

***Abstract*—Before installing a new switch, it is very useful to test the functionality of the switch. Preferably, this is done by a fully automatic program which needs minimal user interaction. In this paper, the design and testoperations are discussed shortly. The implementation of a script or program can be done in several ways and in different languages. In this work, a basic implementation has been made using Peral script, showing the required functionality. Afterwards, a custom benchmark shows if it is useful to implement the same functionality using other, more efficient languages.**
Several communication methodes like serial communication, telnet and SNMP are examined. This paper will prove which communication method is the most effective in a specific situation focussing on getting and setting switch parameters.

I. Introduction

BEFORE configuring and installing new switches at companies, it is recommended to make sure every single ethernet or gigabitport is working properly. Companies are free to sign a staging contract which covers this additional quality test.
At Telindus, the staging process is executed manually.
Not only may this be an extremely lengthy and uninspiring job, but more important, automating these processes also allows to deliver a higher quality service at a lower cost. Concerning these important issues, we wrote a fully automatic script to test Cisco and Juniper switches .
To solve the first issue, the script must ensure minimal user interaction. Other requirements include robustness, speed and universality. This will be discussed in topic **??**.

In the first stage, the most appropriate language has to be chosen. After defining the programming language, the real programming work can be done. While thinking about some useful methods, it became immediately clear that there isn't just one suitable solution. Getting and setting data from and to the switch can be realised with different communication methods.
In this paper, a comparison between serial communication, telnet and SNMP can be found.

Afterwards, another benchmark is set up to decide whether it is useful to reïmplement the script in another, more efficient language.

II. Programming language

Determining the most suitable programming language is the first step taken to realise the script. In the early days, you were restricted to choose between fortran, cobol or lisp. At the moment, the amount of programming languages exceeds the number of thousand!
The need to select some languages to compare is inevitable. This selection can be found below and will be discussed very shortly.

- Java
- C++
- Perl
- Python
- Ruby
- PHP

PHP
PHP is a server-side scripting language. In some applications, it is used to monitor network traffic and display the results in a webbrowser. PHP needs a local or external server to run PHP scripts.

Java
Nortel Device Manager is a GUI tool to configure Nortel switches which is fully written in Java. That's the reason why this language became a promising solution.
Many network applications require multithreading where Java is the ultimate language to handle these multithreaded operations. However, as we will see later, multithreading was not of any interest in our particular situation.

C++
Normally, applications written in C++ are very fast. It's interesting to check if this statement is true regarding network applications.

Perl - Ruby - Python
Unlike Java and C++, these alternatives are scripting languages. Object oriënted programming is possible, especially with Ruby, but it is not it's main purpose.
The syntax of these three languages differ. Ruby and python don't use braces but take care of clarity with tabs. Perl on the other hand uses braces like the most languages do. Some sites ensure that python is the fastest. (http://data.perl.it/shootout) On the other hand, Perl is the fastest along other websites. (http://xodian.net/serendipity/index.php?/archives/27-Benchmark-PHP-vs.-Python-vs.-Perl-vs.-Ruby.html)

The reason for these different results can be easily explained.

Based on one specific benchmark, it would be unfair to conclude that Perl is the fastest in every respect.

It is only possible to compare these languages with a specific purpose in mind. Our purpose is to write a script which automatically tests hardware of a Cisco or Juniper Switch. In this case, it would be useless to benchmark the graphic processing skills of these languages. Testing some network operations would be more effective.
Later on, you will find a custom made benchmark.

III. Communication methods

A. General info

Network programming requires interaction between hosts and network devices such as routers, switches and firewalls. So let's have a look at several communication methods.
Serial communication is mostly used to make a connection through the console port. The greatest advantage is the fact that you are able to establish interaction without the need of any switch configuration. This technique becomes indispensable when neither the ip address, vty ports, console or aux-ports are configured.

The telnet protocol is built upon three main ideas. First, the concept of a 'Network Virtual Terminal'; second, the principle of negotiated options; and third, a symmetric view of terminals and processes. [5]
If multiple network devices are connected to eachother, a client is able to gain remote access to each device which is telnet ready. All information send by telnet, is send in plain text. In this situation, security is not an important issue.

SNMP is a very interesting protcol to get specific info of a device. With one single command, it is possible to retreive the status of an interface, the amount of retreived TCP segments etc.
Three different versions of SNMP are possible

SNMPv1, SNMPv2
SNMP V1 and V2 are very close. They both use community strings to authenticate the packets. The community string is sent in plain-text.
The main difference between V1 and V2 is that SNMPv2 added a few more packet types like the GETBULK PDU which enable you to request a large number of GET or GETNEXT in one packet. Instead of SMIv1, SNMPv2 uses SMIv2 which is a better, with more data types like 64-bit counters, etc... But mostly the difference between V1 and V2 is internal and the end user will probably not notice any difference between the two. [6]

SNMPv3
SNMPv3 was designed to address the weak V1/V2 security. SNMPv3 is more secure than SNMPV2. It does not use community strings but users with passwords, and SNMPv3 packets can be authenticated and encrypted depending on how your users have been defined. In addition, the SNMPv3 framework defines user groups and MIB-views which enable an agent to control the access to its MIB objects. A MIB-view is a subset of the MIB. You can use MIB-views to define what part of the MIB a user can read or write. [6]

B. Benchmark

In this section, we show some figures regarding speed using different possible communication methods (serial communication, telnet and SNMP). Thanks to these benchmarks, we are able to select the most suitable communication method in every case, at every specific moment.
First of all, the benchmark is written in two languages (Perl and Python) to check if the results are not determined by the programming language.
As you can see in figure 4.3 4.4 and 4.5, the relationship between serial, telnet and SNMP is almost the same. At this moment, we can conclude that the results are independent of the programming language.

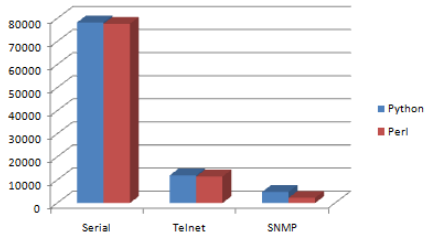


Fig. 1. GET

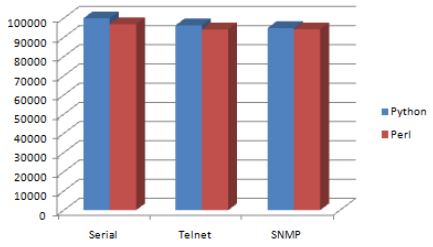


Fig. 2. SET(wait)

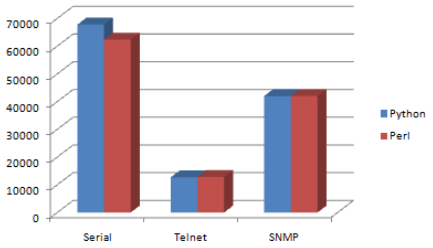


Fig. 3. SET(no wait)

This benchmark is split up into three different tests. Get, Set with a wait function and Set without a wait function.

The length of the command and the execution time of a command are also considered.

GET	Get a variable from the switch. (500 times)
SETwait	Set a parameter of the switch and wait until this parameter is in the requested state. (50 times)
SETnowait	Set a parameter of the switch and it doesn't matter if it is already in the requested state. (500 times)
	Long exectution time - Long command
GET SET	sh interf gigabitEthernet 1/0/1 mtu inter gig 1/0/1 shut
	Long exectution time - Short command
GET SET	sh in gig 1/0/1 mtu in gig 1/0/1 shu
	Short exectution time - Long command
SET	hostname abcdefghij
	Short exectution time - Short command
SET	hostname abc

TABLE I
COMMUNICATION METHODS

Serial GET												
l-l	01:17.921	01:17.187	01:17.265	01:17.546	01:16.921	01:18.500	01:17.687	01:17.671	01:18.750	01:17.140	**77659ms**	σ = 563ms
s-s	00:33.344	00:34.953	00:32.984	00:33.110	00:33.141	00:33.329	00:33.375	00:33.641	00:33.032	00:33.016	**33393ms**	σ = 556ms
Telnet GET												
l-l	00:11.140	00:11.531	00:11.093	00:11.078	00:11.171	00:10.906	00:15.046	00:10.812	00:11.000	00:10.906	**11468ms**	σ = 1207ms
s-s	00:03.844	00:03.266	00:03.578	00:03.469	00:03.359	00:03.469	00:03.438	00:03.390	00:03.297	00:03.406	**3452ms**	σ = 156ms
SNMP GET												
l-l	00:02.562	00:02.312	00:02.062	00:02.187	00:02.277	00:02.043	00:02.168	00:02.183	00:02.355	00:02.248	**2240ms**	σ = 143ms
s-s	00:02.656	00:02.890	00:02.641	00:02.719	00:02.766	00:03.109	00:02.875	00:02.593	00:02.812	00:02.812	**2787ms**	σ = 143ms
Serial SET(wait)												
l-l	01:36.860	01:36.728	01:36.681	01:37.075	01:35.920	01:38.108	01:34.218	01:37.672	01:38.891	01:36.282	**96844ms**	σ = 1210ms
l-s	01:36.110	01:36.343	01:36.374	01:37.611	01:36.788	01:38.656	01:37.131	01:36.625	01:36.335	01:36.140	**96811ms**	σ = 758ms
s-l	00:07.469	00:07.016	00:07.266	00:07.563	00:07.017	00:07.313	00:07.391	00:07.157	00:07.017	00:07.220	**7243ms**	σ = 185ms
s-s	00:05.641	00:06.375	00:05.860	00:06.688	00:05.922	00:06.000	00:06.063	00:05.906	00:06.062	00:05.922	**33393ms**	σ = 278ms
Telnet SET(wait)												
l-l	01:35.048	01:38.954	01:33.673	01:33.298	01:32.967	01:33.827	01:33.717	01:33.171	01:33.546	01:33.406	**94161ms**	σ = 1687ms
l-s	01:34.375	01:33.780	01:35.955	01:34.547	01:34.201	01:33.335	01:32.890	01:34.574	01:36.782	01:33.938	**94438ms**	σ = 1434ms
s-l	00:02.781	00:02.719	00:03.297	00:02.641	00:02.735	00:02.828	00:02.984	00:03.000	00:03.188	00:02.563	**2874ms**	σ = 226ms
s-s	00:02.922	00:02.312	00:03.203	00:02.328	00:02.234	00:02.391	00:02.172	00:02.297	00:02.407	00:02.297	**2456**	σ = 316ms
SNMP SET(wait)												
l-l	01:37.859	01:33.858	01:33.878	01:35.053	01:34.490	01:33.251	01:32.273	01:34.693	01:33.755	01:33.189	**94230ms**	σ = 1431ms
l-s	01:35.425	01:35.374	01:34.577	01:34.375	01:35.519	01:35.594	01:35.955	01:36.250	01:34.688	01:34.780	**95254ms**	σ = 590ms
s-l	00:01.641	00:01.516	00:01.797	00:01.954	00:01.687	00:01.735	00:02.031	00:01.703	00:01.797	00:01.703	**1756ms**	σ = 141ms
s-s	00:01.484	00:01.390	00:01.532	00:01.594	00:01.672	00:01.625	00:01.563	00:01.609	00:01.578	00:01.578	**1562ms**	σ = 75ms
Serial SET(no wait)												
l-l	01:01.985	01:02.563	01:02.110	01:01.735	01:02.750	01:02.735	01:02.703	01:02.360	01:02.016	01:02.485	**62344ms**	σ = 343ms
l-s	00:57.828	00:56.905	00:57.388	00:58.719	00:57.587	00:57.063	00:56.987	00:57.468	00:57.785	00:56.938	**57467ms**	σ = 530ms
s-l	00:51.924	00:50.157	00:51.748	00:50.447	00:50.563	00:50.453	00:50.376	00:50.579	00:51.125	00:50.821	**50819ms**	σ = 566ms
s-s	00:41.922	00:41.579	00:41.344	00:42.407	00:41.016	00:40.453	00:41.903	00:41.343	00:42.104	00:41.187	**41526ms**	σ = 548ms
Telnet SET(no wait)												
l-l	00:12.531	00:13.109	00:12.468	00:12.719	00:12.625	00:14.281	00:12.563	00:12.594	00:12.610	00:12.547	**12805ms**	σ = 520ms
l-s	00:10.703	00:10.890	00:10.687	00:10.484	00:10.515	00:10.890	00:10.781	00:10.484	00:10.734	00:10.672	**10684ms**	σ = 143ms
s-l	00:12.171	00:12.109	00:12.672	00:12.640	00:12.484	00:13.172	00:12.594	00:12.422	00:12.891	00:12.703	**12586ms**	σ = 299ms
s-s	00:09.328	00:09.156	00:09.157	00:09.828	00:09.016	00:09.063	00:09.266	00:09.250	00:09.047	00:09.094	**9221ms**	σ = 225ms
SNMP SET(no wait)												
l-l	00:42.906	00:42.031	00:41.875	00:41.968	00:41.984	00:41.809	00:41.582	00:42.082	00:42.734	00:41.766	**42074ms**	σ = 399ms
l-s	00:43.483	00:42.701	00:42.014	00:44.532	00:44.751	00:43.543	00:43.832	00:42.609	00:42.986	00:43.014	**43347ms**	σ = 815ms
s-l	00:43.811	00:43.687	00:44.312	00:43.687	00:43.544	00:41.844	00:43.206	00:41.578	00:44.057	00:41.969	**43169ms**	σ = 930ms
s-s	00:42.657	00:41.642	00:42.157	00:41.642	00:41.860	00:41.860	00:42.578	00:41.795	00:41.781	00:41.624	**41960ms**	σ = 361ms

TABLE II
RESULTS (CFR PDF)

discussion of the results

Figure 4.3, 4.4 and 4.5 represent the relationship between serial communication, telnet and SNMP(left graphs). It also shows the influence of the command length and the duration of the execution time (right graphs).

GET operation

SNMP is the best communication method to get information of the switch. Telnet can be used as well when the commands are short. It is recommended to avoid serial communication.
The first step taken to explain these differences is to take a glance at the overhead.
The speed of serial communication is 9600 bps and has 2 bit overhead to 8 bits data. This is the start and stop bit. A parity bit is not used in this test. Telnet packets flow at a higher speed (100Mbps in this situation). The speed gain is less than 100 000 000 / 9.600 because telnet has more overhead. To send 1 frame, telnet needs 90 bytes. Another difference is the protocol being used. Telnet uses TCP while SNMP uses UDP. That's why SNMP has to deal with less overhead (66 bytes / frame). Every command is small enough to fit in just one frame. So the overhead is not the main reason for these speed differences. The fact that TCP is connection oriënted and UDP is connection less should be a better explanation. TCP takes care of acknowledging every octet. This is done by seq and ack flags which slows down the communication.
Concerning the length of a command, we expect that serial communication and telnet are faster because less data has to be sent. In this example when shorter commands are used, telnet becomes 3.32 times faster. Serial communication speeds up too, but only 2.32 times. Serial communication needs 2 extra bits to send 1 byte. Telnet doesn't need extra bits because one byte can be encapsulated in the same frame. SNMP seems not to be influenced that much by command length because an SNMP get-request consists of an object identifier witch contains almost the same size.
The benchmark shows that SNMP is faster than telnet. A difference in waiting time will be an additional explanation. SNMP doesn't need to wait for the prompt, while telnet and serial communication have to cope with this waiting time.

SET(wait) operation

Imagine a programmer must shut down an interface before another interface may come up. It takes some time when the interface is up and running. To make sure the interface is in the right state, the programmer must wait until the previous operation is ready. This execution time differs from command to command. Shutting down an interface takes more time than setting the hostname.
In this situation, when the execution time is high, the choice of communication method is not that important. The waiting time will be the bottleneck. When the execution time is low, the speed in descending order is SNMP, telnet and serial communication. The reason can be found in previous section. Sometimes, telnet will be prefered because SNMP does not support every set command.

SET(no wait) operation

While configuring a switch, it is not necessary to wait until the previous command is really executed. Note that you still need to wait for the prompt.
It is remarkable that SNMP is not the fastest anymore and this communication method is not influenced by command length and execution time. After an SNMP set request is send, an SNMP get response is received when the command is really executed. So SNMP is slower because it checks automatically if the command is executed well.
Serial communication comes close to SNMP when we have to deal with short commands. Telnet is the obvious victor. The reason is already mentioned above.
At this point, we are able to decide which communication method is the most efficient in a particular situation.

	Telnet	SNMP	Serial
Datalink (ethernet)	38	38	startbit
Network (IPv4)	20	20	+
Transport	32(TCP)	8(UDP)	stopbit
Total	90 bytes	66 bytes	2 bit

TABLE III
OVERHEAD

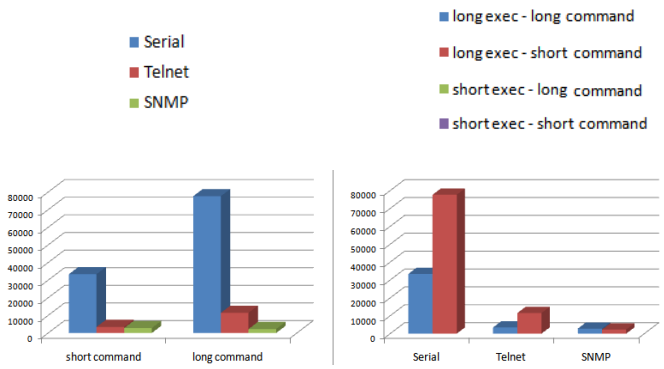


Fig. 4. GET

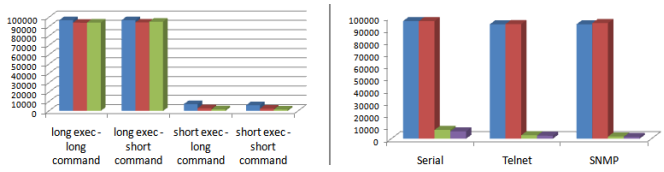


Fig. 5. SET(wait)

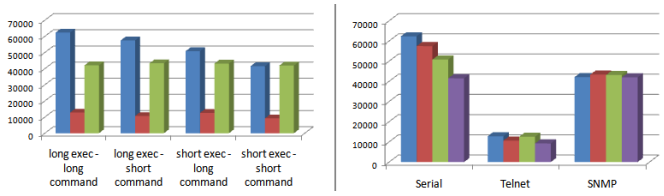


Fig. 6. SET(no wait)

IV. Script

As previously mentioned, a script would be very useful to test a Cisco or Juniper switch automatically. Some conditions must be met. The script must be fast, robust, universal and needs minimal user interaction.
This section describes the operation of the script.

A. Purpose

Before a switch will be installed at a company, this script will prove that every interface is able to send and receive data. If no errors are detected, the switch has passed the test, which can be verified in a HTML report showing every detected error. The possibility to add some configuration automatically is an extra useful feature. The switch can be tested and configured at the same time.

B. Design

The script will need an FTP server, a PC from which the script has run, a MasterSwitch and a SlaveSwitch. The

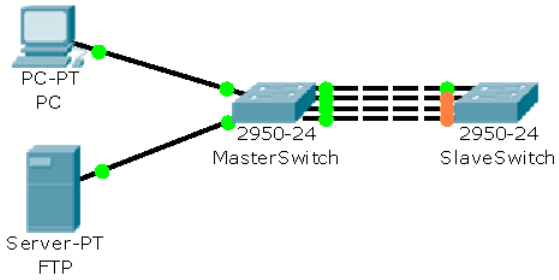


Fig. 7. Design

SlaveSwitch is the switch being tested.
There are several ways to connect these components. The most suitable wiring can be found in figure **??**.
This design provides a universal solution to test a standalone Cisco or Juniper switch and a Cisco chassis with supervisor installed. It is possible to eliminate the external FTP server by using flash memory of the switch as a directory for an FTP transfer. Note this implies some disadvantages. Enough space on the flash is required and this solution is not that universal for Cisco and Juniper switches.

As you see, critical connections are attached directly to the MasterSwitch. Critical connections are connections from which you have to be 100% sure they are operational. In this case, it's the link between MasterSwitch - PC and MasterSwitch - FTP. The other connections are for testing purpose. This increases the reliability of the test. On the other hand, programming becomes more complex. The programmer has to deal with vlan's to redirect icmp and tcp packets to the SlaveSwitch.

C. Test operations

The purpose of the script can be summarized into one sentence. Testing each interface on errors to make sure you can install the switch in an operational environment. It is possible to test the interfaces at different levels. It would be possible to check if the bit error rate for a given operational time does not exceed the treshhold. To accomplish this, it is necessary to send a huge amount of data. If you send 1 kB, it is not sufficient to observe the BER. This kind of test is not suitable because the script needs to be fast.
A second approach is to check the functionality of the interfaces. A successful ping guarantees the interface is responding. This test does not ensure that the specific interface is capable to transport an amount of data from or to another interface without any errors. Therefor, an FTP transfer will be used.

D. Flowchart test operations

Vlan's are necessary because data has to travel through the SlaveSwitch. Below, you find the vlan scheme and the corresponding traffic flow.

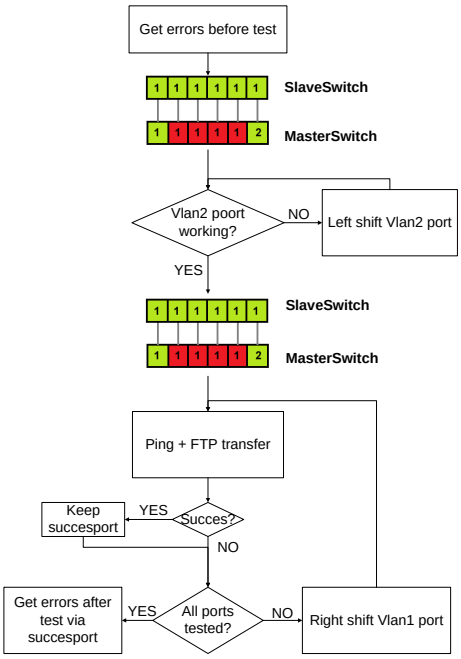


Fig. 8. Flowchart Testoperations

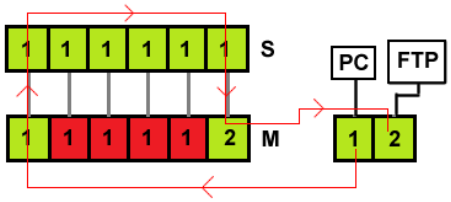


Fig. 9. VLAN configuration

V. Custom made benchmark

After the script is written, it is useful to check which language is the most appropriate among those languages which are discussed at the beginning of this paper. Looking at the result, we consider whether or not to rewrite the script. To accomplish this, we designed a custom made benchmark.
We counted every operation which is executed during the script. For example, if an SNMP request is done, a counter iSNMP is added by 1. The next step taken is to eliminate some negligible operations such as split functions. They were only executed 5 times. The remaining results can be found in figure 12.1.

Then, all these operations needs to be programmed in Java, C++, Perl, Python, Ruby and PHP. Each operation is executed as many times as seen in 'Quantity of executions'.
To accomplish operations like SNMP requests, sometimes external modules / packages are used. A list of all used packages can be found in table 12.3.

Note that implementation-inefficiency is dealt with. Here is the explanation using an example.
During a telnet connection, it is necessary to wait for the prompt before sending a new command. Some modules or packages already contains this wait command. Mostly, they use a sleep command for a specific period which is extremely unefficient. We wrote our own wait function similar for every language. Is this wait function written as fast as possible? Probably yes, but if not, it will not influence the result because each languages uses this function.
Another example is the ping command. It is possible to add options to the ping command like the number of echo requests and the time-out time. Each language uses the same options especially 4 echo requests and 3000 ms time-out time. An ICMP ping is used instead of a TCP or UDP ping.

Type	Amount	Percent	Quantity of executions (500000 measurements)
Regex	93802	0,665744013	332872
Variable changes	40713	0,288953711	144477
Function calls	2848	0,020213204	10107
SNMP	1687	0,0119732	5987
If functions	1099	0,007799969	3900
Push array	674	0,004783602	2392
Ping	26	0,000184531	92
FTP transfer	24	0,000170336	85
Telnet operations	25	0,000177433	89

TABLE IV
USED OPERATIONS

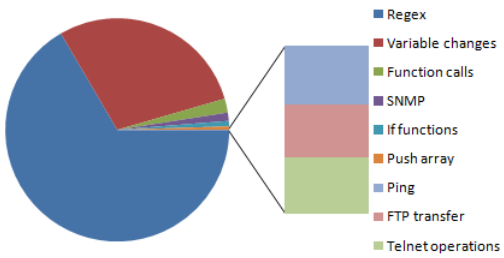


Fig. 10. Used operations

	System specifications
PC	HP Compaq NC 6120 (1.86GHz, 2GB RAM)
Platform	Windows XP (32 bit)
	interpreter/compiler
Perl	ActivePerl 5.10.1.1007
Python	Python 2.6.4
Ruby	Ruby 1.9.1-p376
Java	JDK 6u19 and NetBeans 6.8
C++	Visual C++ 2008 Express Edition
PHP	WampServer 2.0i with PHP 5.3.0
	used packages
Perl	[4][5][6][7]
Python	[8][9][10][11][12]
Ruby	[13][14][15][16]
Java	[17][18][19]
C++	[20][21]
PHP	[22]

TABLE V
REQUIREMENTS

Table 12.4 shows the result of the benchmark. 10 results / language are measured to minimize effects caused by coincidence. Not only speed, but also memory usage and page faults have been taken into account. The latter two are not mentioned because no significant differences could be found. Java needs more memory, but nowadays memory became very cheap.

Perl	01:45.546	01:43.796	01:48.546	01:43.889	01:44.780	01:42.093	01:40.705	01:42.515	01:46.440	01:43.906	**104182 ms**	σ = 2142ms
Ruby	02:21.156	02:18.593	02:20.296	02:20.530	02:15.999	02:17.943	02:26.088	02:18.831	02:15.408	02:33.437	**140828 ms**	σ = 5070ms
Python	02:10.171	02:10.296	02:09.467	02:04.624	02:11.671	02:07.874	02:22.264	02:10.780	02:14.608	02:08.186	**130994 ms**	σ = 4497ms
PHP	03:00.454	02:58.308	02:57.960	03:03.256	03:01.936	02:59.375	03:02.162	03:10.087	03:03.672	02:58.644	**181585 ms**	σ = 1320ms
Java	01:32.326	01:31.530	01:38.607	01:30.510	01:33.558	01:34.546	01:32.474	01:33.643	01:41.844	01:36.428	**94547 ms**	σ = 3312ms
C++	01:43.425	01:41.096	01:39.315	01:38.329	01:39.565	01:41.426	01:38.567	01:37.238	01:38.642	01:37.939	**99554 ms**	σ = 1794ms

TABLE VI
RESULTS (CFR PDF)

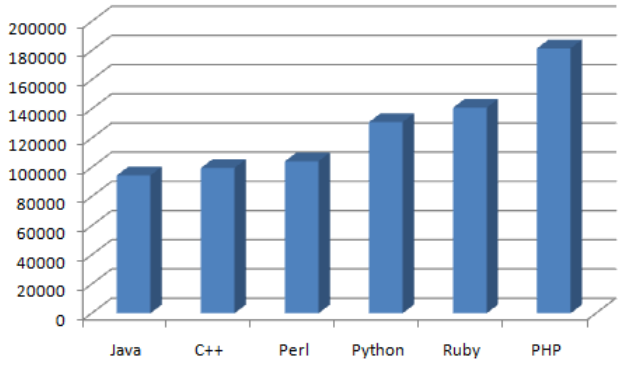


Fig. 11. benchmark results

As you can see in figure 12.2, it can be easily seen that Perl is the fastest among all scripting languages. As mentioned before, it's a good idea to wonder whether it is useful to rewrite the script in Java or C++. Let's take a look at the results. Perl needs 104182ms to handle the script. C++ and Java are respectively 4.442% and 9.248% faster. Because all operations are executed approximately 3.56 times the original value, these percentages will be strongly reduced. We can conclude that rewriting the script doens't give a remarkable additional value.

VI. Conclusion

To test a switch manually, it takes about 16 minutes 8 seconds. Thanks to the script, a switch can be tested in 2 minutes 41 seconds. To accomplish this improvement, we benchmarked three different communication methodes. When SNMP is prefered in one case, telnet or serial communication are recommended in another. Table 13.1 offers you a short summary. the 'x' represents a don't care. If two options are mentioned, the first one is the most desirable. Keeping these results in mind, the script is written in Perl. Afterwards, a custum made benchmark constatates that rewriting the script doens't give a remarkable additional value. Perl is the best among all scripting languages. This language also provides some effective external modules to handle network operations. Java and C++ are the fastest, but requires better programming skills.
From now on, this script will be in use at Telindus headquarters.

GET		
execution time	**command length**	
x	long	SNMP / Telnet
x	short	SNMP / Telnet
SET wait		
long	long	x
long	short	x
short	long	SNMP / Telnet
short	short	SNMP / Telnet
SET no wait		
long	long	Telnet
long	short	Telnet
short	long	Telnet
short	short	Telnet

TABLE VII
CONCLUSION

Acknowledgment

We would like to express our gratitude to Dirk Vervoort, Kristof Braeckman, Jonas Spapen and Toon Claes for their technical support. We also want to thank Staf Vermeulen and Niko Vanzeebroeck for supervising the entire master thesis process. Also thanks to Joan De Boeck for his scientific assistance.

References

[1] *Net-SNMP-v6.0.0*, Available at http://search.cpan.org/dist/Net-SNMP/
[2] *Net-Ping-2.36*, Available at http://search.cpan.org/ smpeters/Net-Ping-2.36/lib/Net/Ping.pm
[3] *Net-Telnet-3.03*, Available at http://search.cpan.org/ jrogers/Net-Telnet-3.03/lib/Net/Telnet.pm
[4] *libnet-1.22*, Available at http://search.cpan.org/ gbarr/libnet-1.22/Net/FTP.pm
[5] *Regular expression operations*, Available at http://docs.python.org/library/re.html#module-re
[6] *pysnmp 0.2.8a*, Available at http://pysnmp.sourceforge.net
[7] *ping.py*, Available at http://www.g-loaded.eu/2009/10/30/python-ping/
[8] *telnetlib*, Available at http://docs.python.org/library/telnetlib.html
[9] *ftplib*, Available at http://docs.python.org/library/ftplib.html
[10] *SNMP library 1.0.1*, Available at http://snmplib.rubyforge.org/doc/index.html
[11] *Net-Ping 1.3.1*, Available at http://raa.ruby-lang.org/project/net-ping/
[12] *Net-Telnet*, Available at http://ruby-doc.org/stdlib/libdoc/net/telnet/rdoc/classes/Net/Telnet.html
[13] *Net-FTP*, Available at http://ruby-doc.org/stdlib/libdoc/net/ftp/rdoc/index.html
[14] *SNMP4j v1/v2c*, Available at http://www.snmp4j.org/doc/index.html
[15] *telnet package*, Available at http://www.jscape.com/sshfactory/docs/javadoc/com/jscape/summary.html
[16] *SunFtpWrapper*, Available at http://www.nsftools.com/tips/SunFtpWrapper.java
[17] *ASocket.h,ASocket_i.c,ASocketConstants.h*, Available at ftp://ftp.activexperts-labs.com/samples/asocket/Visual%20C++/Include/
[18] *Regular expressions*, Available at http://msdn.microsoft.com/en-us/library/system.text.regularexpressions.aspx
[19] *PHP telnet 1.1*, Available at http://www.geckotribe.com/php-telnet/
[20] Philip M. Miller, *TCP/IP - The Ultimate Protocol Guide*, BrownWalker Press, 2009
[21] Cisco Press, *CNAP CCNA 1 & 2 Companion Guide Revised (3rd Edition)*, Cisco systems, 2004
[22] Douglas R Mauro, Kevin J Schmidt,*Essential SNMP 2nd Edition*, O'Reilly Media, 2005
[23] Charles Spurgeon, Ethernet: The Definitive Guide, O'Reilly and Associates, 2000

[1]IBW, K.H. Kempen (Associatie KULeuven), Kleinhoefstraat 4, B-2440 Geel, Belgium

[2]Telindus nv, Geldenaaksebaan 335, B-3001 Heverlee, Belgium